Evaluation analytique de la précision des systèmes en virgule fixe pour des applications de communication numérique

Aymen Chakhari

ANNÉE 2014

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : TRAITEMENT DU SIGNAL ET TÉLÉCOMMUNICATION

Ecole doctorale MATISSE

présentée par

Aymen CHAKHARI

préparée à l'unité de recherche CAIRN, IRISA, INRIA
INSTITUT NATIONALE DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE
ENSSAT LANNION

Evaluation analytique de la précision des systèmes en virgule fixe pour des applications de communication numérique

Thèse à soutenir à Lannion le 7 Octobre 2014

devant le jury composé de :

Kosai RAOOF
Professeur, ENSIM Le Mans / rapporteur

Christophe JEGO
Professeur, ENSEIRB Bordeaux / rapporteur

Meriem JAIDANE
Professeur, U2S, ENIT Tunis / examinateur

Daniel MENARD
Professeur, IETR INSA Rennes / examinateur

Pascal SCALART
Professeur, ENSSAT Lannion / directeur de thèse

Romuald ROCHER
Maître de conférences, IRISA / co-directeur de thèse

Résumé

Lors de la conception d'applications de traitement de signal, la démarche classique conduit le concepteur d'applications à utiliser dans un premier temps une arithmétique en virgule flottante pour éviter les problèmes liés à la précision des calculs. Cependant, l'implémentation de ces applications nécessite l'utilisation de l'arithmétique virgule fixe puisqu'elle est plus avantageuse en termes de contraintes de coût et de consommation. Par conséquent l'application conçue en arithmétique virgule flottante doit être convertie en arithmétique virgule fixe. Cette opération de conversion se révèle être fastidieuse, aussi des outils de conversion automatique de l'arithmétique virgule flottante vers celle en virgule fixe ont été mis en place afin de répondre aux exigences de temps de mise sur le marché de ces applications. Au sein de ce processus de conversion, l'une des étapes primordiales concerne l'évaluation de la précision de la spécification en virgule fixe. En effet, le changement du format des données de l'application s'effectue en éliminant des bits. Cette conversion conduit à la génération de bruits de quantification qui se propagent au sein du système et dégradent la précision des calculs en sortie de l'application. Par conséquent, cet abaissement dans la précision de calcul doit être maitrisé et évalué pour garantir l'intégrité de l'algorithme et répondre aux spécifications initiales de l'application. Traditionnellement, l'évaluation de la précision s'effectue par le biais de deux approches différentes. La première approche consiste à réaliser des simulations en virgule fixe de l'application afin d'en estimer ses performances. Ces approches basées sur la simulation nécessitent de grandes capacités de calculs et conduisent à des temps d'évaluation prohibitifs. Pour éviter ce problème, le travail mené dans le cadre de cette thèse se concentre sur des approches basées sur l'évaluation de la précision à travers des modèles analytiques. Ces modèles décrivent le comportement du système à travers des expressions analytiques qui permettent d'évaluer une métrique de précision bien définie. Plusieurs modèles analytiques ont été proposés pour évaluer la précision en virgule fixe des systèmes linéaires invariants dans le temps LTI (*Linear Time Invariant*) et des systèmes linéaires non-LTI non récursifs et récursifs. L'objectif de cette thèse est de proposer des modèles analytiques permettant l'évaluation de la précision des systèmes de communications numériques et des algorithmes de traitement numérique de signal formés d'opérateurs non lisses et non linéaires en terme du bruit. Dans un premier temps, des modèles analytiques pour l'évaluation de la précision des opérateurs de décision et leurs cascades et itérations sont proposés. Dans une seconde étape, une optimisation des largeurs de données est proposée pour l'implémentation matérielle en virgule fixe de l'égaliseur à retour de décision DFE (*Decision Feedback Equalizer*) en se basant sur les modèles analytiques proposés ainsi que pour les algorithmes de décodage itératif du type turbo-décodage et décodage-LDPC (*Low-Density Parity-Check*) selon une loi se quantification particulière.

Le premier aspect de ce travail concerne la proposition de modèles analytiques pour l'évaluation de la précision des opérateurs non lisses de décision et de la cascade des opérateurs de décision. Par conséquent, la caractérisation de la propagation des erreurs de quantification dans la cascade d'opérateurs de décision est le fondement des modèles analytiques proposés. Ces modèles sont appliqués dans une seconde étape pour l'évaluation de la précision de l'algorithme de décodage sphérique SSFE (*Selective Spanning with Fast Enumeration*) utilisé pour les systèmes de transmission de type MIMO (*Multiple-Input Multiple-Output*). Dans une seconde étape, l'évaluation de la précision des structures itératives d'opérateurs de décision a fait l'objet d'intérêt. Une caractérisation des erreurs de quantification engendrées par l'utilisation de l'arithmétique en virgule fixe est introduite pour aboutir à des modèles analytiques permettant l'évaluation de la précision des applications de traitement numérique de signal incluant des structures itératives de décision. Une deuxième approche, basée sur l'estimation d'une borne supérieure de la probabilité d'erreur de décision dans le mode de convergence, est proposée pour l'évaluation de la précision de ces applications et ceci dans un but de réduire les temps d'évaluation. Ces modèles sont appliqués à la problématique de l'évaluation de la spécification virgule fixe de l'égaliseur à retour de décision DFE. Une extension de ces modèles est introduite pour l'évaluation du DFE dans sa version adaptative.

Le second aspect de notre travail s'articule autour de l'optimisation des largeurs de données en virgule fixe. Ce processus d'optimisation est basé sur la minimisation de la probabilité d'erreur de décision dans le cadre d'une implémentation sur un FPGA (*Field-Programmable Gate Array*) de l'algorithme DFE complexe sous contrainte d'une précision donnée. Par conséquent, pour chaque spécification en virgule fixe, la précision est évaluée à travers les modèles analytiques proposés. L'estimation de la consommation des ressources et de la puissance sur le FPGA est ensuite obtenue à l'aide des outils de Xilinx pour faire un choix adéquat des largeurs des données en visant à un compromis précision/coût.

La dernière étape de notre travail concerne la modélisation en virgule fixe des algorithmes de décodage itératif. Une modélisation de l'algorithme de turbo-décodage et du décodage LDPC est ensuite proposée. Cette approche intègre la structure particulière de ces algorithmes ce qui implique que les quantités calculées au sein du décodeur ainsi que les opérations, sont quantifiées suivant une approche itérative. De plus, la représentation en virgule fixe utilisée est différente de la représentation classique utilisant le nombre de bits accordé à la partie entière et la partie fractionnaire. L'approche proposée repose sur le couple dynamique et le nombre de bits total. De plus le choix de la dynamique engendre davantage de flexibilité pour les modèles en virgule fixe puisqu'elle n'est plus limitée uniquement à une puissance de deux. Dans une seconde étape, la réduction de la taille des mémoires par des techniques de saturation et de troncature est proposée afin d'être en mesure de cibler des architectures à faible-complexité. Finalement, l'analyse des performances en virgule fixe est faite à travers l'évaluation du taux d'erreur par paquet FER (*Frame Error Ratio*) en fonction du SNR (*Signal to Noise Ratio*).

Abstract

In designing applications of signal processing, the traditional approach leads the designer of applications to use initially floating point arithmetic in order to avoid problems related to the accuracy of calculations. However, the implementation of these applications requires the use of fixed-point arithmetic because it is more advantageous in terms of constraints of cost and consumption. Therefore, the designed application in floating point arithmetic must be converted to fixed-point arithmetic. This conversion is tedious, so tools for automatic conversion of floating-point arithmetic to the fixed point were established to meet the requirements of time-to-market of these applications. In this conversion process, one of the basic steps concerns the evaluation of the accuracy of the fixed-point specification. Indeed, the change of the data format of the application is performed by removing bits. This conversion results in the generation of quantization noise propagating within the system and degrading the accuracy of calculations of the application output. Therefore, this reduction in the calculation accuracy must be mastered and evaluated in order to ensure the integrity of the algorithm and meet the initial requirements of the application. Traditionally, evaluation of accuracy is performed through two different approaches. The first approach is to perform simulations fixed-point implementation in order to assess its performance. These approaches based on simulation require large computing capacities and lead to prohibitive time evaluation. To avoid this problem, the work done in this thesis focuses on approaches based on the accuracy evaluation through analytical models. These models describe the behavior of the system through analytical expressions that evaluate a defined metric of precision. Several analytical models have been proposed to evaluate the fixed-point accuracy of Linear Time Invariant systems (LTI) and of non-LTI non-recursive and recursive linear systems. The objective of this thesis is to propose analytical models to evaluate the accuracy of digital communications systems and algorithms of digital signal processing made up of non-smooth and non-linear operators in terms of noise. In a first step, analytical models for evaluation of the accuracy of decision operators and their iterations and cascades are provided. In a second step, an optimization of the data length is given for fixed-point hardware implementation of the Decision Feedback Equalizer DFE based on analytical models proposed and for iterative decoding algorithms such as turbo decoding and LDPC decoding-(Low-Density Parity-Check) in a particular quantization law.

The first aspect of this work concerns the proposition analytical models for evaluating the accuracy of the non-smooth decision operators and the cascading of decision operators. So, the characterization of the quantization errors propagation in the cascade of decision operators is the basis of the proposed analytical models. These models are applied in a second step to evaluate the accuracy of the spherical decoding algorithm SSFE (Selective Spanning with Fast Enumeration) used for transmission MIMO systems (Multiple-Input Multiple - Output). In a second step, the accuracy evaluation of the iterative structures of decision operators has been the interesting subject. Characterization of quantization errors caused by the use of fixed-point arithmetic is introduced to result in analytical models to evaluate the accuracy of application of digital signal processing including iterative structures of decision. A second approach, based on the estimation of an upper bound of the decision error probability in the convergence mode, is proposed for evaluating the accuracy of these applications in order to reduce the evaluation time. These models are applied to the problem of evaluating the fixed-point specification of the Decision Feedback Equalizer DFE. An extension of these models is introduced for the evaluation of DFE in its adaptive version.

The second aspect of our work focuses on the optimization of fixed-point data widths. This optimization process is based on minimizing the decision error probability through the implementation on an FPGA (Field-Programmable Gate Array) of the complex DFE algorithm under the constraint of a given accuracy. Therefore, for each fixed-point specification, accuracy is evaluated through the proposed analytical models. The estimation of resources and power consumption on the FPGA is then obtained using the Xilinx tools to make a proper choice of the data widths aiming to a compromise accuracy/cost.

The last step of our work concerns the fixed-point modeling of iterative decoding algorithms. A model of the turbo decoding algorithm and the LDPC decoding is then given. This approach integrates the particular structure of these algorithms which implies that the calculated quantities in the decoder and the operations are quantified following an iterative approach. Furthermore, the used fixed-point representation is different from the conventional representation using the number of bits accorded to the integer part and the fractional part. The proposed approach is based on the dynamic and the total number of bits. Besides, the dynamic choice causes more flexibility for fixed-point models since it is not limited to only a power of

two. In a second step, the memory size reduction using saturation and truncation techniques is given in order to be able to target low - complexity architectures. Finally, the fixed-point performance analysis is done through the evaluation of Frame Error Ratio FER versus SNR (Signal to Noise Ratio) package.

Table des matières

Table des figures

Liste des tableaux

Glossaire

A	Arrondi
AC	Arrondi Convergent
APA	Algorithmes de Projection Affine
ASIC	Application Specific Integrated Circuit
ASIP	Application Specific Instruction-set Processor
CA2	Complément à 2
DSP	Digital Signal Processor
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
LMS	Least Mean Square
LSB	Last Significant Bit
MCR	Moindres Carrés Récursifs
MSB	Most Significant Bit
NLMS	Normalize Least Mean Square
RLS	Recursive Least-Square
RSBQ	Rapport Signal sur Bruit de Quantification
T	Troncature
TNS	Transmission Numérique du Signal

Notations

E	Terme d'espérance
q	Pas de quantification
$x(n)$	Donnée scalaire à un instant n
$X(n)$	Donnée vectorielle à un instant n
$\mathbf{X}(n)$	Donnée matricielle à un instant n
m_X	Moyenne de la donnée $X(n)$ scalaire ou matricielle
μ_x	Moyenne du scalaire $x(n)$
σ_x^2	Variance du scalaire $x(n)$
$\mathbf{R}_{XX}(\theta) = E[X(n)X^t(n-\theta)]$	Matrice d'autocorrélation des données $X(n)$
$\mathbf{C}_{XX}(\theta) = E\left[(X(n) - m_X)\left(X^t(n-\theta) - m_X^t\right)\right]$	Matrice de covariance des données $X(n)$

Introduction

De nos jours, le Traitement Numérique de Signal ne cesse d'évoluer pour répondre aux exigences actuelles de plusieurs domaines tels que les télécommunications qui ont marqué une avancée impressionnante. De plus, les algorithmes de Traitement Numérique de Signal figurent dans différentes applications telles que le domaine médical, l'aéronautique, le multimédia et l'électronique grand public. L'évolution rapide de la technologie des semi-conducteurs a conduit à d'énormes innovations dans le domaine de conception des produits électroniques pour implanter des applications de plus en plus complexes. Les équipements électroniques disponibles sur les marchés d'aujourd'hui sont caractérisés par une concurrence entre les grandes entreprises d'électronique pour les rendre plus efficaces et multifonctionnels. Cela s'est manifesté par l'évolution rapide des réseaux radio mobile particulièrement avec la quatrième génération (ou 4G). Par conséquent, les équipements de ces normes intègrent différentes technologies de communication numérique (WIFI, Bluetooth, GPS,...) qui cohabitent ensemble au sein du même équipement avec plusieurs applications fondées sur des algorithmes de traitement numérique de signal. En conséquence, ces grandes firmes d'électronique se trouvent face à un cycle continu de conception et de livraison des produits. A chaque itération de ce cycle, les concepteurs ont besoin de travailler avec une très haute complexité technologique en visant en même temps à fournir davantage de valeur ajoutée au consommateur dans les meilleurs délais.

Les *smart-phones* actuels avec la $4G$ sont un bon exemple d'un équipement électronique de télécommunication, de traitement de signal et de semi-conducteurs et qui nécessite de très hautes capacités de calcul pour accomplir les diverses applications qui sont embarquées dans ces équipement et assurer les performances demandées pour la voix et le trafic des données. De plus, ils sont équipés de plusieurs outils de multimédia tels que les caméras par exemple pour enregistrer des vidéos et jouer de la musique en temps réel. Tout cela est conçu sous les contraintes d'énergie et de temps. L'exemple des *smart-phones* peut être généralisé pour la conception des équipements électroniques modernes afin de à minimiser le coût du système en termes de surface de silicium, le profit de la consommation de puissance et le temps d'exécution sous contrainte de performances améliorées en termes de précision de calcul et du temps de réponse. Par conséquent, ces objectifs représentent "pour les concepteurs" un compromis entre les performances et le coût. En conséquence, il est très important de faire les choix adéquats à chaque étape de conception afin d'assurer les meilleurs performances possibles du système global. De plus, le temps de mise sur le marché (*Time-to-market*) de ces produits doit être réduit. Pour cette raison, la conception et le développement de ces applications nécessitent l'utilisation des outils de haut-niveau qui interviennent dans toutes les étapes du cycle de développement des applications de traitement numérique de signal et de communication numérique. Le rôle de ces outils est de fournir aux concepteurs des facilités d'implantation pour choisir la meilleure architecture à implémenter en permettant de passer d'un haut niveau d'abstraction à une caractérisation bas niveau. Également, le choix des opérateurs utilisés pour implémenter ces algorithmes influence très largement le compromis coût/performance. Les opérateurs en arithmétique virgule flottante et en arithmétique virgule fixe sont les deux choix les plus courants pour l'implémentation de toutes les opérations arithmétiques. Il convient de préciser que l'implémentation en arithmétique virgule fixe possède l'avantage d'une consommation de puissance inférieure à celle en arithmétique virgule flottante, mais aussi elle est marquée par une latence plus courte et des surfaces significativement plus petites. Par conséquent l'arithmétique en virgule fixe est un choix populaire pour l'implémenta-

tion des algorithmes de traitement numérique de signal (TNS) qui possèdent des paramètres de performances rigoureux à achever et qui demandent des puissances de calcul élevées.

Contexte de l'étude

Le processus de conception des algorithmes de traitement numérique de signal s'effectue selon plusieurs étapes. Le concepteur est ainsi amené à faire le choix approprié en considérant plusieurs options disponibles à chaque étape du cycle de conception de l'application. Un choix particulier qui conduit à l'amélioration des performances du système est invariablement associé à l'augmentation indésirable du coût du système. Par conséquent, la prise de décision d'un bon choix est réalisé par l'intermédiaire d'un nombre d'itérations de vérification avant d'arriver au choix optimal.

Les différentes étapes de conception d'une application de traitement numérique de signal sont décrites par la figure 1.

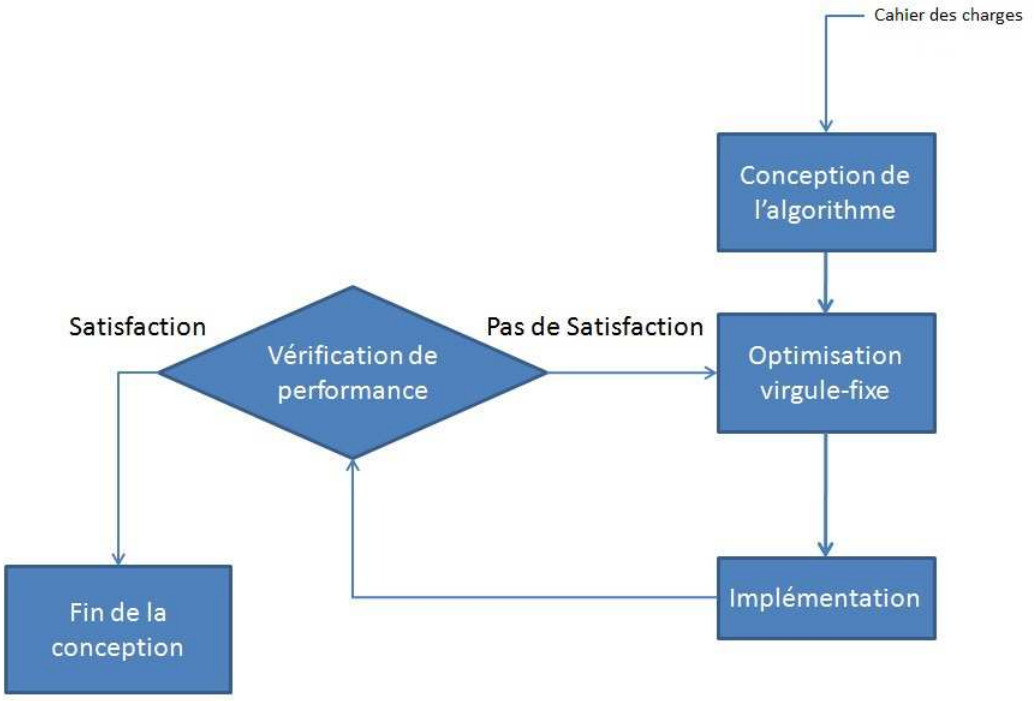

FIGURE 1 – Cycle de conception d'une application de traitement numérique de signal

La première étape de ce cycle est la définition du cahier de charge en spécifiant les différentes fonctions à accomplir par le système conçu ainsi que les différentes contraintes architecturales du système par rapport à la consommation d'énergie et au niveau temporel. L'étape suivante correspond à la conception de l'algorithme et la spécification d'une description complète de l'algorithme qui est une tâche donnée dans le contexte de cette thèse. A cette étape, des simulations sont effectuées pour vérifier les différents critères conçus à travers des outils tels que Matlab (Mathworks), Scilab (Inria). Ces simulations sont effectuées en utilisant l'arithmétique virgule flottante pour s'affranchir des problèmes de précision des calculs. Une fois l'algorithme est conçu, celui ci est décrit à l'aide d'un langage de programmation (C,C embarqué) et implanté dans le système embarqué. L'implantation peut être logicielle ou matérielle à travers les solutions architecturales tels que ASIC, FPGA ou DSP. L'étape suivante concerne la détermination d'une représentation convenable en virgule fixe. Cette tache est le centre d'intérêt de cette thèse. Les contraintes de coût et de consommation des systèmes embarqués conduisent à l'utilisation de l'arithmétique en virgule fixe puisqu'elles sont moindres par rapport à l'arithmétique virgule flottante car les largeurs des données de cette dernière sont plus larges qu'en format virgule fixe.

Généralement, les données en virgule flottante sont codées sur 32 bits alors qu'en virgule fixe, elle sont codées sur 16 bits. Également, les opérateurs en virgule fixe sont moins complexes qu'en virgule flottante. Par contre, le temps de développement en virgule fixe est plus important car les formats des différentes données en virgule fixe doivent être optimisés et bien définis pour éviter le débordement au sein de l'application. Par conséquent une conversion de la logique virgule flottante vers l'arithmétique virgule fixe est effectuée. Cette tâche, longue, peut constituer 30% du temps d'implantation du système. De plus, elle conduit à de nombreuses erreurs et ainsi la représentation en virgule fixe doit garantir l'intégrité de l'application et satisfaire les contraintes de performance. Afin de répondre aux exigences du temps de mise sur le marché, cette conversion est automatisée à travers des outils de conversion automatique qui visent à répondre aux exigences de la précision pour maintenir les performances de l'algorithme tout en optimisant les coûts. Une fois qu'un format convenable de virgule fixe est sélectionné, le système est implémenté et une vérification des performances du système selon les critères de l'utilisateur est effectuée. Si les résultats sont satisfaisants, la conception est clôturée, sinon, toutes les étapes à partir de l'optimisation en arithmétique virgule fixe doivent être répétées pour compenser les lacunes du format virgule fixe précédent.

Le bloc d'implémentation n'est pas détaillé dans la figure 1 afin de mettre l'accent sur l'élément central de cette thèse. Dans la pratique, ce bloc est complexe et peut être séparé en plusieurs sous-blocs. Dans le cas d'une implémentation logicielle, il existe des compilateurs, tels que le compilateur GCC (*GNU Compiler Collection*), qui sont spécifiquement optimisés pour des architectures de processeurs spécifiques.

Problématique de la thèse

Il existe quatre tâches pour l'implémentation en arithmétique virgule fixe d'un système donné tel que montré par la figure 2.

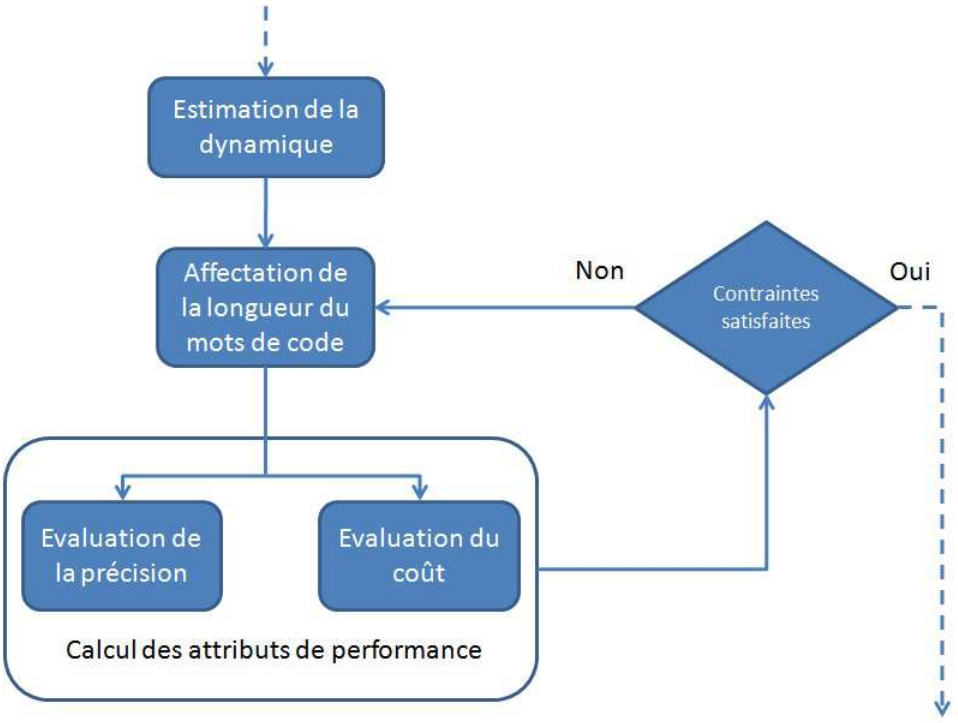

FIGURE 2 – Les étapes d'implémentation en virgule fixe

La première tâche concerne l'estimation de la dynamique des signaux. En effet, une détermination de la dynamique est requise pour connaître l'intervalle des valeurs prises par les données.

Ensuite le nombre de bits accordé à la partie entière du format virgule fixe est déduit. En d'autres termes, cette première étape conduit à déterminer la position de la virgule qui doit permettre d'éviter la présence de débordements. En addition, un alignement de la virgule des opérandes est effectué par le biais des opérations de recadrage afin d'adapter les données à leurs dynamiques. Dans une seconde étape, l'optimisation en arithmétique virgule fixe nécessite de déterminer le nombre de bits à affecter à la partie fractionnaire. Ce nombre de bits résulte d'un compromis entre le coût d'implantation et la précision des calculs.

La deuxième tâche concerne la mesure des pertes en précision causées par l'utilisation des opérateurs et des données en arithmétique virgule fixe. En effet, l'utilisation de l'arithmétique virgule fixe engendre des erreurs de calculs par rapport aux opérations réalisées en précision infinie. Par conséquent, l'évaluation de la précision des calculs est une étape primordiale dans le processus de conception des applications en format virgule fixe. Cette tâche constitue l'objet principal de ce travail de thèse, en particulier l'évaluation de la précision des systèmes de traitement numérique de signal et de communication numérique constitués d'opérateurs non lisses (*un-smooth*) et non linéaires en terme du bruit. L'évaluation de la précision se fait par le biais d'une métrique caractérisant les performances de la précision qui est généralement le Rapport Signal à Bruit de Quantification (RSBQ). Dans [82], une méthode est présentée pour déterminer la valeur minimale du RSBQ et par conséquent l'optimisation en arithmétique virgule fixe se fait sous une contrainte de précision obtenue par le biais de la valeur minimale du RSBQ. De plus, l'estimation des avantages du coût correspondant à l'utilisation du format virgule fixe constitue un défi à résoudre. En effet, l'objectif visé est de faire un compromis entre les pertes en précision et les gains en coût d'implémentation en respectant les contraintes de la conception et en faisant un choix optimisé du format virgule fixe.

Dans le cas d'une implémentation logicielle sur un DSP ou bien un micro-contrôleur, l'architecture est fixée et les opérateurs sont prédéfinis sur une largeur bien précise. Néanmoins, les processeurs peuvent manipuler plusieurs types d'instructions telles que les instructions de type SWP (*Sub-Word Parallelism*) qui permettent d'accélérer les temps d'exécution en favorisant le parallélisme des opérations. D'autres part, les processeurs fournissent des instructions multiprécisions permettant de traiter les données enregistrées en mémoire avec une précision plus élevée. Le temps d'exécution T_{exe} de ce type d'instruction est alors plus important et le problème posé consiste à trouver le point optimum de fonctionnement correspondant au temps minimal d'exécution T_{exe} satisfaisant également la contrainte de précision $RSBQ_{min}$. En d'autres termes, le temps d'exécution T_{exe} est minimisé tant que la contrainte de précision $RSBQ_{min}$ est satisfaite. Cette approche conduit au problème d'optimisation suivant :

$$\min_{B \in E_B} (T_{exe}(B)) \qquad tel \quad que \qquad RSBQ(B) \geq RSBQ_{min} \tag{1}$$

où B représente le vecteur contenant les largeurs des données présentes dans l'algorithme et E_B désigne l'ensemble des largeurs supportées par le processeur. Le temps d'exécution du code dépend du type d'instruction choisi.

L'implémentation matérielle sur un FPGA ou un ASIC est fondée sur une définition de l'architecture qui détermine le nombre d'opérateurs impliqués dans l'application. Particulièrement, la largeur des différents opérateurs arithmétiques est à fixer. Dans ce cas d'implémentation, l'objectif de la méthodologie est de minimiser le coût de l'architecture en termes de consommations de ressources et d'énergie sous contrainte d'une précision donnée des calculs. Les coûts locaux des éléments présents dans l'architecture conduisent à la minimisation du coût global C_{arch} de l'architecture sous la contrainte de précision minimale exprimée par $RSBQ_{min}$ (équation 2). Une telle approche fera l'objet du troisième chapitre de cette thèse dédié à l'implémentation efficace sur FPGA de l'algorithme d'égalisation à retour de décision sous contrainte d'une probabilité d'erreur de décision minimale.

$$\min_{B \in E_B} (C_{arch}(B)) \qquad tel \quad que \qquad RSBQ(B) \geq RSBQ_{min} \tag{2}$$

Les 2 méthodologies définies par les relations 1 et 2 font appel à une contrainte d'optimisation liée à l'évaluation de la précision des calculs en arithmétique virgule fixe. L'évaluation de cette précision constitue la pierre angulaire de ce travail de thèse. Deux approches différentes sont classiquement envisagées pour effectuer cette tâche : l'approche basée sur des simulations en arithmétique virgule fixe et l'approche basée sur des modèles analytiques permettant de caractériser la métrique de précision. Étant donné que la précision des calcul est évaluée à chaque itération du processus d'optimisation, l'approche par simulation engendre des temps d'optimisation prohibitifs. Par conséquent, pour limiter cet effet, des modèles analytiques ont été proposés pour évaluer la précision en virgule fixe des systèmes de communication numérique et de traitement numérique de signal. Dans ce dernier cas, la métrique de précision (RSBQ(B)) est calculée à chaque itération du processus ce qui nécessite la connaissance de son expression analytique en chaque point du graphe décrivant l'algorithme. Par conséquent, le temps d'évaluation est significativement réduit.

Dans la littérature, des approches analytiques ([32], [69]) ont été proposées pour évaluer la précision des systèmes linéaires et invariants dans le temps (LTI), les systèmes non-LTI non-récursifs dans [70] et également les systèmes LTI et non-LTI récursifs. L'objectif de ce travail de recherche est de proposer des modèles analytiques des systèmes de communication numérique et de traitement numérique de signal formés par des opérateurs non lisses et non linéaires en terme de bruit tels que les algorithmes de décodage sphérique, l'égalisation à retour de décision ou encore le turbo-décodage. De plus, des optimisations des largeurs des données en format virgule fixe pour ces applications ont été abordées en se basant sur les modèles analytiques proposés.

Contributions

Les travaux effectués durant cette thèse s'articulent autour trois grands piliers. Dans un premier temps, des modèles analytiques pour l'évaluation de la précision des systèmes constitués d'une cascade d'opérateurs non lisses de décision ont été proposées en collaboration avec les travaux de recherche effectués dans le cadre de la thèse de Karthick PARSHAR au sein de la même équipe CAIRN. Cette étude repose sur l'identification d'opérateurs non lisses et sur la caractérisation des bruit de quantification engendrés par ce type d'opérateurs. Dans une seconde étape, ce travail a été étendu à l'analyse du phénomène de propagation des erreurs de décision lorsque plusieurs opérateurs de décision sont utilisés en cascade. Ce travail nous a permis de définir un modèle analytique général permettant d'évaluer la précision de systèmes constitués d'une cascade d'opérateurs de décision comme, par exemple, les systèmes de communications numériques. Ainsi, les modèles analytiques proposés seront appliqués à l'évaluation des algorithmes de décodage sphérique tel que l'algorithme SSFE (*Selective Spanning with Fast Enumeration*).

La seconde contribution de cette thèse concerne l'évaluation de la précision des systèmes de communication numérique basés sur l'itération d'opérateurs non lisses de décision. Cette étude a fait l'objet d'une approche analytique permettant l'évaluation de la précision en arithmétique virgule fixe lorsque l'itération d'opérateurs non lisses de décision converge en se basant sur la résolution d'un système d'équations non linéaires par la méthode de Newton-Raphson. Cependant, le temps d'évaluation de cette approche demeurant assez élevé pour des constellations à forte efficacité spectrale, nous proposons une approche alternative dans une seconde étape afin de remédier à ce problème. Cette dernière fournit une borne supérieure de la probabilité d'erreur de décision lorsque les systèmes convergent. Elle sera appliquée à l'évaluation de la précision de l'égaliseur à retour de décision. Enfin, la dernière contribution de cette partie concerne l'optimisation de l'implémentation en virgule fixe sur FPGA d'un égaliseur numérique à retour de décision ou DFE (Deision Feedback). Cette optimisation est effectuée sous une contrainte de précision exprimée à travers la probabilité d'erreur de décision. Ainsi, une architecture de DFE a été proposée permettant de fixer les largeurs des données de cet algorithme à partir d'un compromis entre la consommation en ressources matérielles et en énergie obtenue pour différentes contraintes de précision.

Nous avons proposé une architecture en format virgule fixe du turbo décodeur qui utilise une

loi de quantification basée sur le couple (A, N_x) où A représente la dynamique de signal et N_x le nombre total de bits utilisés dans la représentation en virgule fixe du signal x à quantifier. Nous montrerons que ce schéma de quantification est plus flexible que la quantification classique représentée par le couple du nombre de bits pour la partie entière et le nombre de bits pour la partie fractionnaire (N_I, N_F), sans tenir compte de la dynamique. En effet, dans les approches classiques, la dynamique est généralement limitée à une puissance de deux tandis que cette approche s'affranchit de cette restriction. Dans une seconde étape, nous avons visé à réduire la taille des mémoires utilisées dans l'implémentation du turbo décodeur afin de satisfaire la contrainte de faible complexité. Cette réduction est faite à travers des techniques de saturation et de troncature. Finalement, une évaluation des performances en arithmétique virgule fixe est réalisée par l'analyse des taux d'erreur par paquet FER (*Frame Error Rate*) en fonction du SNR (*Signal to Noise Ratio*). Des choix sur la largeur des données sont discutés afin de réaliser un compromis entre les solutions à hautes performances et celles à faibles complexités. L'ensemble de cette méthodologie a également été appliqué à la modélisation d'un décodeur LDPC en se basant sur le principe de fonctionnement du décodage-LDPC.

Organisation de la thèse

Cette thèse est constituée de quatre parties. Le premier chapitre est un chapitre introductif présentant les différents aspects des arithmétiques en virgule flottante et en virgule fixe avec une comparaison entre les deux arithmétiques en termes de plusieurs critères. Cette comparaison justifie l'intérêt de l'utilisation de la représentation en virgule fixe pour les systèmes embarqués et les algorithmes de traitement numérique de signal. Ensuite, les différentes approches pour l'évaluation de la précision en virgule fixe sont explicitées, en particulier les approches par simulation et les approches basées sur des modèles analytiques. Une comparaison entre ces deux approches est ensuite présentée ce qui conduira à justifier l'avantage de l'utilisation des modèles analytiques en termes d'optimisation du temps d'évaluation et des capacités de calcul.

Le deuxième chapitre de cette thèse définit tout d'abord la notion d'opérateurs non lisses en proposant une technique pour les identifier. Nous présentons ensuite un modèle analytique pour l'évaluation de la précision d'un opérateur de décision non lisse. Dans une seconde étape le problème de la propagation des erreurs de décisions dans une cascade d'opérateurs de décision est abordé en proposant des modèles analytiques permettant l'évaluation de ce genre de configuration dans les systèmes de communication numérique. Ces modèles analytiques ont été appliqués à l'algorithme SSFE.

La troisième partie de cette thèse est consacrée à la présentation des modèles analytiques permettant l'évaluation de la précision des itérations d'opérateurs de décision. En premier lieu, une étude est menée sur la caractérisation du phénomène de propagation des erreurs causées par l'opérateur de quantification utilisé dans une structure itérée. Deux modèles analytiques sont ensuite présentés pour évaluer la probabilité d'erreur de décision dans le mode de convergence. Ces modèles sont appliqués, dans une seconde étape, pour évaluer la précision de l'algorithme d'égalisation à retour de décision DFE tout en présentant une solution pour l'algorithme DFE adaptatif. La dernière partie de ce chapitre présente une optimisation en arithmétique virgule fixe des largeurs de données du DFE pour une implémentation sur un FPGA. Cette optimisation vise à minimiser le coût de l'architecture en termes de consommation des ressources et de puissance sous une contrainte de précision exprimée sous forme de probabilité d'erreur de décision.

Le dernier chapitre de cette thèse présente des modélisations en arithmétique virgule fixe de deux algorithmes de décodage itératif. Nous présentons tout d'abord le principe du turbo décodage. Nous proposons ensuite un modèle en arithmétique virgule fixe pour ce décodeur en se basant sur des techniques de saturation et de troncature pour viser à des solutions à faible complexité. La dernière partie du chapitre est consacrée à l'évaluation expérimentale du modèle proposé à travers la mesure des performances en FER. De même, nous présentons dans la dernière partie le principe de décodage LDPC en proposant une architecture en format virgule fixe de

ce dernier. L'évaluation des performances en arithmétique virgule fixe du modèle proposé est effectuée à travers les résultats obtenus.

Chapitre 1

Arithmétique virgule fixe

Sommaire

Cette partie est consacrée à la présentation des spécifications de l'arithmétique virgule fixe ainsi que des différentes techniques existantes pour évaluer la précision des systèmes virgule fixe. En effet, les arithmétiques virgule fixe et virgule flottante sont les arithmétiques binaires de données les plus utilisées sur les plates-formes informatiques modernes. La première partie de ce chapitre est consacrée à la description des différents types de codage. Les circuits en arithmétique virgule fixe ont existé depuis les premiers ordinateurs électroniques et les calculateurs. Choisir le format approprié en arithmétique virgule fixe a toujours été un sujet de préoccupation. Auparavant, ce choix était influencé principalement par des considérations liées à la puissance de calcul des processeurs mais également par des considérations sur la mémoire nécessaire en raison de la latence et du coût de celle-ci. La plupart des algorithmes numériques complexes devaient compter sur des tables précalculées durant les calculs. Aujourd'hui, la définition du format en virgule fixe d'un nombre est impactée par des considérations telles que la surface de silicium, le temps d'exécution et la consommation totale d'énergie du dispositif électronique. L'arithmétique virgule fixe répond bien à ces exigences par rapport à l'arithmétique flottante. Par contre, l'utilisation de l'arithmétique virgule fixe engendre la génération de bruits de quantification causés par les débordements lors du changement de format. Ces bruits, présentés dans la deuxième partie de ce chapitre, se propagent ensuite au sein du système global et conduisent à la modification de la qualité de l'application. Par conséquent, l'évaluation de la précision est une étape primordiale lors de la conversion de format flottant au format fixe afin de garantir l'intégrité de l'application. Il

est donc important de disposer de méthodologies d'évaluation de cette précision en arithmétique virgule fixe. Ce point fera l'objet de la troisième partie de ce chapitre. Il existe deux familles d'approches : l'approche d'évaluation par simulation qui est basée sur des simulations virgule fixe de l'application et l'approche d'évaluation analytique qui consiste à trouver des modèles analytiques basés sur des métriques. Nous menons ci-après une analyse des points forts de chacune de ces deux approches ce qui nous permettra de sélectionner l'approche retenue dans le cadre de ce travail.

1.1 Les différents types de formats de données

Le choix d'une représentation binaire relève d'une problématique qui concerne toute plateforme numérique actuelle. Faire un choix d'une représentation ou d'une arithmétique représente une étape primordiale dans le développement d'une application numérique. Les arithmétiques virgule fixe et virgule flottante sont généralement utilisées dans un but de stockage ou de calcul. En général, les algorithmes de traitement numérique de signal demandent des capacités de calcul intensif. Par conséquent, le choix du bon format de représentation des nombres joue un rôle crucial pour une implémentation efficace de n'importe quel algorithme de traitement numérique de signal.

Dans le calcul numérique, le système de numération spécifie la méthode selon laquelle les nombres sont représentés comme étant une séquence de chiffres binaires et il spécifie également les règles pour effectuer les opérations arithmétiques (ex. addition, multiplication etc.) entre ces nombres. Dans la plupart du temps, les calculs scientifiques donnent seulement une approximation de la valeur exacte qui serait obtenue avec une précision infinie. Ceci est une conséquence du nombre limité de bits utilisés dans le système de numération. Quel que soit l'arithmétique virgule fixe ou l'arithmétique virgule flottante utilisée, seulement un nombre fini de bits est utilisé pour la représentation des nombres réels.

La précision limitée des standards de codage peut être évaluée selon deux perspectives différentes. La première concerne la précision des calculs déterminée par l'étape de quantification du système de numération (la distance entre deux nombres). Le second aspect est relatif à la variation de la dynamique maximale permise par la représentation. Cette variation de la dynamique d'un système de numération est donnée par l'ensemble des valeurs possibles qui peuvent être représentées. Elle est évaluée à travers le logarithme du quotient entre les amplitudes maximale et minimale du signal comme indiqué à l'équation (1.1).

Par conséquent, la comparaison entre le codage en virgule flottante et en virgule fixe est fondée en particulier sur l'analyse de la précision numérique et la variation de la dynamique selon la relation :

$$D_{dB} = 20log_{10}\left(\frac{X_{MAX}}{X_{MIN}}\right). \tag{1.1}$$

Généralement, les algorithmes pour les systèmes embarqués sont développés en utilisant l'arithmétique virgule flottante afin d'éviter les problèmes liés à la longueur finie d'un mot de code. Ce processus valide l'intégrité de l'algorithme et vérifie si l'algorithme proposé répond bien au cahier des charges. Même si l'erreur inhérente à l'exactitude de calcul existe toujours, elle demeure faible par rapport à celle obtenue par l'arithmétique en virgule fixe. Par conséquent, le calcul en virgule flottante garantit une précision et une variation de dynamique suffisantes dans la plupart des cas.

Néanmoins, la plupart des implémentations VLSI utilisent l'arithmétique en virgule fixe pour réduire la surface et la consommation d'énergie et obtenir un matériel rentable. En contrepartie, une dégradation de la précision des calculs est produite en raison du nombre limité de bits utilisés dans la représentation des données. En effet, l'utilisation du codage en virgule fixe introduit des bruits de quantification lors de l'élimination de bits à travers les opérations de saturation (arrondi) et troncature. En outre, cela conduit à l'apparition de débordements à chaque fois que la longueur du mot de code de la partie entière est insuffisante pour représenter l'ensemble de la variation de la dynamique.

Dans cette section, les différents types d'arithmétiques pour les systèmes numériques sont abordés. En premier lieu, l'arithmétique virgule flottante est introduite, ensuite, l'arithmétique virgule fixe est présentée en la comparant avec celle virgule flottante afin de montrer les avantages et les inconvénients de chacune de ces deux représentations.

1.1.1 Représentation des nombres entiers

Numération simple de position

Un entier p (compris entre 0 et $B^N - 1$) est décomposé dans la base B d'une façon unique selon la formule suivante :

$$p = \sum_{i=0}^{N-1} a_i B^i. \tag{1.2}$$

Les a_i sont des chiffres compris entre 0 et $B - 1$. L'entier p est noté dans la base B de la façon suivante : $(a_{N-1}...a_2a_1a_0)_B$ ou tout simplement $(a_{N-1}...a_2a_1a_0)$ si la base B a été prédéfinie auparavant. Ainsi la chaîne $a_{N-1}...a_2a_1a_0$ est nommée l'entier simple associé à p en base B (ESA_B) selon :

$$ESA_B(a_{N-1}...a_2a_1a_0) = \sum_{i=0}^{N-1} a_i B^i. \tag{1.3}$$

Notation Signe - Valeur Absolue (SVA)

La notation en numération simple d'un nombre signé correspond à la représentation de la valeur absolue de ce nombre en numération simple de position en ajoutant un symbole spécifiant le signe du nombre à représenter. L'inconvénient de cette représentation se manifeste par les opérations mathématiques effectuées sur des nombres respectant cette notation. En effet, à titre d'exemple, l'addition de deux nombres nécessite l'utilisation de deux algorithmes : un algorithme d'addition si les deux nombres possèdent le même signe et un algorithme de soustraction dans le cas contraire.

Présentation en complément à la base

Afin d'éviter le problème de prise en compte du signe des opérandes présent pour la notation SVA, le système de numération des entiers, en complément à la base B a été créé. Par conséquent dans le cas où B est une base paire, l'entier $p = (a_{N-1}...a_2a_1a_0)_{\bar{B}}$ est représenté de la façon suivante :

- Si $0 \leq p \leq B^N/2 - 1$, la représentation est celle de l'équation (1.2) :
$$ESA_B(a_{N-1}...a_2a_1a_0) = p.$$
- Si $-B^N/2 \leq p \leq -1$, le codage est réalisé selon l'expression suivante :
$$ESA_B(a_{N-1}...a_2a_1a_0) = B^N + p.$$

Par conséquent, grâce à cette notation, on utilise un seul algorithme d'addition modulo B^N pour réaliser l'addition de tous les entiers compris entre $-B^N/2$ et $B^N/2$. Si le signe du résultat de l'addition est différent de celui des opérandes, il en résulte un dépassement de capacité. Dans le cas où $B = 2$, cette notation est connue sous le nom de notation à *complément à deux* (CA2).

1.1.2 Le format virgule fixe

Soit x un réel écrit sous la forme $x = pK$ avec p un entier codé selon l'une des représentations décrites précédemment où K est appelé le facteur d'échelle. Ce facteur d'échelle est une puissance de la base B préalablement fixée : $K = B^n$. La position de la virgule peut alors être référencée par un entier n relativement au bit de poids faible (*Least Significant Bit* ou LSB) ou par un entier m relativement au bit de poids fort (*Most Significant Bit* ou MSB). Dans ce cas, le réel x peut se représenter selon la figure 1.1 où les deux entiers n et m représentent les distances entre la virgule et le LSB et le MSB (*Most Signifiant Bit*) respectivement.

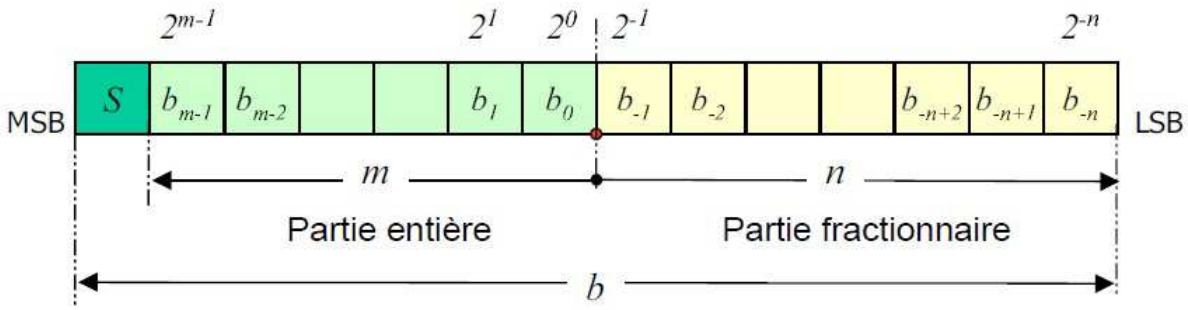

FIGURE 1.1 – Représentation des données en virgule fixe

La figure 1.1 représente un codage en virgule fixe d'un nombre avec un bit de signe S, m bits pour la partie entière et n bits pour la partie fractionnaire. Le nombre de bits utilisés dans cette représentation est $b = m + n + 1$.

Le format d'une donnée en virgule fixe est entièrement défini par la représentation choisie et la largeur de sa partie entière et de sa partie fractionnaire. Généralement, les nombres en arithmétique virgule fixe utilisent une représentation en complément à deux. La valeur d'un nombre est donnée par la relation suivante :

$$x = -2^m S + \sum_{i=-n}^{m-1} b_i 2^i. \tag{1.4}$$

Cette représentation possède des propriétés arithmétiques intéressantes pour les opérations d'addition et soustraction. En effet, même si les résultats intermédiaires d'une série d'additions sont en dehors du domaine de définition du codage, le résultat final sera correct si celui-ci appartient au domaine de définition du codage. De plus, l'implantation dans les processeurs numériques des opérateurs traditionnels utilisant ce code est simplifiée lorsque l'on utilise cette représentation.

$$x = -2^m S + \sum_{i=-n}^{m-1} b_i 2^i. \tag{1.5}$$

Par ailleurs, elle possède l'avantage de n'autoriser qu'une seule représentation possible pour 0. Par conséquent, le domaine des valeurs possibles n'est pas symétrique par rapport à l'origine, et il possède 2^{m+n} valeurs négatives et $2^{m+n} - 1$ valeurs positives, soit :

$$\mathcal{D} = [-2^m; 2^m - 2^{-n}]. \tag{1.6}$$

La notation en format virgule fixe est simple et permet d'effectuer les calculs rapidement. En contrepartie, comme le facteur d'échelle K est fixé, la notation en virgule fixe ne permet pas de représenter des nombres d'ordre de grandeur très différents. De plus, il n'est pas toujours facile de fixer *a priori* le facteur d'échelle si l'ordre de grandeur des valeurs prises par la donnée n'est pas connu par avance. La notation en virgule flottante permet de résoudre ces problèmes.

1.1.3 Le format virgule flottante

Le système des nombres en virgule flottante est le standard de codage le plus utilisé lorsqu'une grande précision est requise. Il représente un nombre réel selon une notation scientifique avec une partie fractionnaire appelée *mantisse* et un facteur d'échelle appelé *exposant*, comme représenté dans la figure 1.2. L'exposant est défini comme une puissance de la base (typiquement 2 et 10) et il est utilisé comme un facteur d'échelle explicite qui change durant les calculs permettant ainsi

une dynamique large pour les valeurs représentées. La mantisse détermine la précision de nombre représenté.

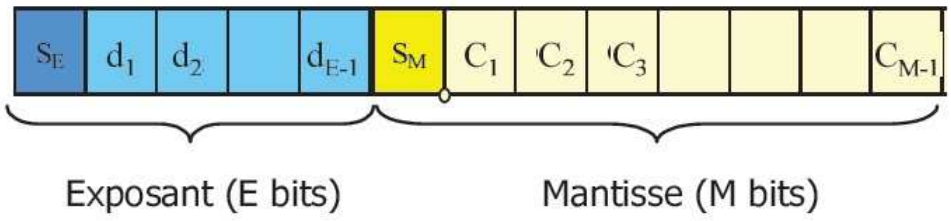

FIGURE 1.2 – Représentation des données en virgule flottante

Soit x un réel signé écrit en format flottant dans la base B avec un signe S, une mantisse M et un exposant E. La valeur associée à ce réel est donnée par l'équation suivante :

$$x = (-1)^S M B^E. \tag{1.7}$$

Comme le nombre de bits accordés à la mantisse et à l'exposant peut prendre un grand nombre de valeurs possibles, un format virgule flottante normalisé a été introduit. La norme IEEE pour l'arithmétique virgule fixe binaire (IEEE 745-2008) est utilisée dans la plupart des CPUs d'aujourd'hui. Elle spécifie le format virgule flottante et également les modes d'arrondis. Elle décrit non seulement comment les opérations arithmétiques doivent être effectuées, mais aussi le traitement des exceptions (division par zéro, les débordements,...). La valeur d'un nombre représenté selon le format virgule flottante de la norme IEEE 745 est calculée :

$$x = (-1)^S 1M 2^{E-biais}. \tag{1.8}$$

La mantisse est normalisée pour représenter une valeur dans l'intervalle $[1, 2]$. Par conséquent, la valeur de son premier bit est fixée à 1 et devient implicite. La valeur de l'exposant est codée comme un nombre non signé. Alors, afin de représenter les nombres inférieurs à 1, un biais est introduit. Ce biais dépend du nombre de bits alloués à l'exposant : $biais = 2^{N_e-1} - 1$. Dans le cas d'un format de précision simple, le biais est 127 et la gamme de l'exposant est $[-126, 127]$. Pour la double précision, le biais est 1023 et l'exposant prend ses valeurs dans $[-1022, 1023]$.

Il est à noter que la représentation d'un nombre n'est pas unique et les zéros à gauche de la mantisse dégradent la précision. Par exemple, considérons la base 10 avec 4 chiffres de mantisse, π peut être représenté par 0.03110^2 ou 3.14210^0. Ces deux représentations ne possèdent pas la même précision. Par conséquent, pour éviter cette ambiguïté, une représentation sans zéro à gauche est proposée et correspond à la représentation normalisée en virgule flottante. Dans ce cas, le premier bit de la mantisse représente le coefficient $\frac{1}{2}$ et est fixé à 1. La valeur de ce bit est inchangée au cours du traitement et ne figure pas dans dans la représentation.

Dans le cas où l'exposant est une puissance de deux (cas binaire), la mantisse et l'exposant sont codés avec une représentation en signe valeur absolue, la valeur de la donnée x est la suivante :

$$x = 2^u (-1)^{S_M} \Big(\frac{1}{2} + \sum_{i=1}^{M-1} C_i 2^{-i-1}\Big) \quad \text{avec} \quad u = (-1)^{S_E} \sum_{i=1}^{E-1} d_i 2^i. \tag{1.9}$$

La valeur 0 n'est pas représentable selon l'équation (1.9). Par conséquent, le domaine de définition des valeurs représentables avec ce codage est l'union de deux sous-domaines donnés par l'expression suivante :

$$\mathcal{D}_R = \Big[-2^K; -2^{-K-1}\Big] \bigcup \Big[2^{-K-1}; 2^K\Big] \quad \text{avec} \quad K = 2^{E-1} - 1. \tag{1.10}$$

L'écart minimal entre deux valeurs représentables par le code considéré détermine le pas de quantification dont la valeur varie en fonction de la valeur représentée. Pour les valeurs de x comprises dans l'intervalle $[-2^u, -2^{u-1}] \bigcup [2^u, 2^{u-1}]$, le pas de quantification est égal à :

$$q = 2^u 2^{-(M+1)}. \tag{1.11}$$

L'étape de quantification de la norme de codage en virgule flottante dépend de la valeur à représenter. En effet, lorsque la valeur de l'exposant augmente (respectivement diminue), la distance entre deux nombres successifs représentables devient plus importante (respectivement plus faible), i.e. le pas de quantification augmente (respectivement diminue) conformément à la relation d'ordre :

$$2^{-(M+1)} < \frac{q}{\mid x \mid} < 2^{-M}. \tag{1.12}$$

Ainsi, la présence d'un facteur d'échelle explicite permet d'adapter le pas de quantification à la valeur de la donnée à coder.

1.1.4 Comparaison arithmétiques virgule flottante et virgule fixe

Nous présentons dans cette section une comparaison entre le format virgule flottante et le format virgule fixe afin de mettre en évidence leurs avantages et inconvénients. Cette comparaison est fondée sur l'analyse de deux aspects. Le premier est l'aspect arithmétique et le deuxième est le coté implantation matérielle puisque l'objectif de cette thèse concerne l'évaluation de la précision et la recherche du format optimal de représentation conduisant à une implémentation matérielle efficace en termes de consommation d'énergie et de ressources.

Comparaison d'un point de vue arithmétique

La qualité du codage est analysée, d'un point de vue arithmétique, en évoquant la dynamique de chacun des deux codages et également le rapport signal à Bruit de quantification.

Comparaison de la dynamique Une représentation efficace des nombres doit à la fois supporter une dynamique large mais également disposer d'une grande précision. En d'autres termes, la représentation d'un nombre devient plus polyvalente avec un rapport plus élevé de la dynamique et la précision. Typiquement, les deux formats virgule fixe et virgule flottante sont symétriques par rapport à leur espace des valeurs représentables. Donc, la dynamique dont l'expression est donnée par l'équation (1.1) est utilisée pour mesurer la fidélité du format. Plus la dynamique est importante, plus le débordement est faible. Dans le cas de la représentation en virgule flottante la dynamique s'exprime par l'équation (1.13) :

$$D_{dB} \simeq 20 \log 10(2^{2K+1}) \quad \text{avec} \quad K = 2^{N_e - 1} - 1. \tag{1.13}$$

N_e représente le nombre de bits accordés à l'exposant. Pour un format en simple précision où l'exposant est représenté par 8 bits, la dynamique devient :

$$D_{dB} = 20 \log 10(2^{(2^8-1)}) = 20 log10(2^{255}) \simeq 1535 \quad dB. \tag{1.14}$$

Pour le format virgule fixe, la dynamique varie linéairement en fonction du nombre de bits, b, utilisé pour la représentation :

$$D_{dB} = 20 log10 \left(\frac{X_{MAX}}{X_{MIN}} \right) = 20 log10 \left(2^{b-1} \right) dB. \tag{1.15}$$

En appliquant les propriétés logarithmiques et des simplifications, la relation précédente devient :

$$D_{dB} = 20 \left(b - 1 \right) log10(2) \simeq 6.02 \left(b - 1 \right). \tag{1.16}$$

La variation de la dynamique en fonction de la longueur du mot de code est plus importante pour les nombres en virgule flottante que pour les nombres en arithmétique virgule fixe. A titre d'exemple, dans le tableau 1.1 le format en précision simple est comparé avec différents types de représentation des données en format virgule fixe. Une différence importante entre les différents formats peut être observée. En effet, le format virgule fixe sur 128 bits possède significativement une petite dynamique par rapport à la représentation virgule flottante sur 32 bits.

Format	Dynamique (dB)
Précision simple	1535
virgule fixe 16 bits	90
virgule fixe 32 bits	186
virgule fixe 64 bits	379
virgule fixe 128 bits	764

TABLE 1.1 – Comparaison de la dynamique

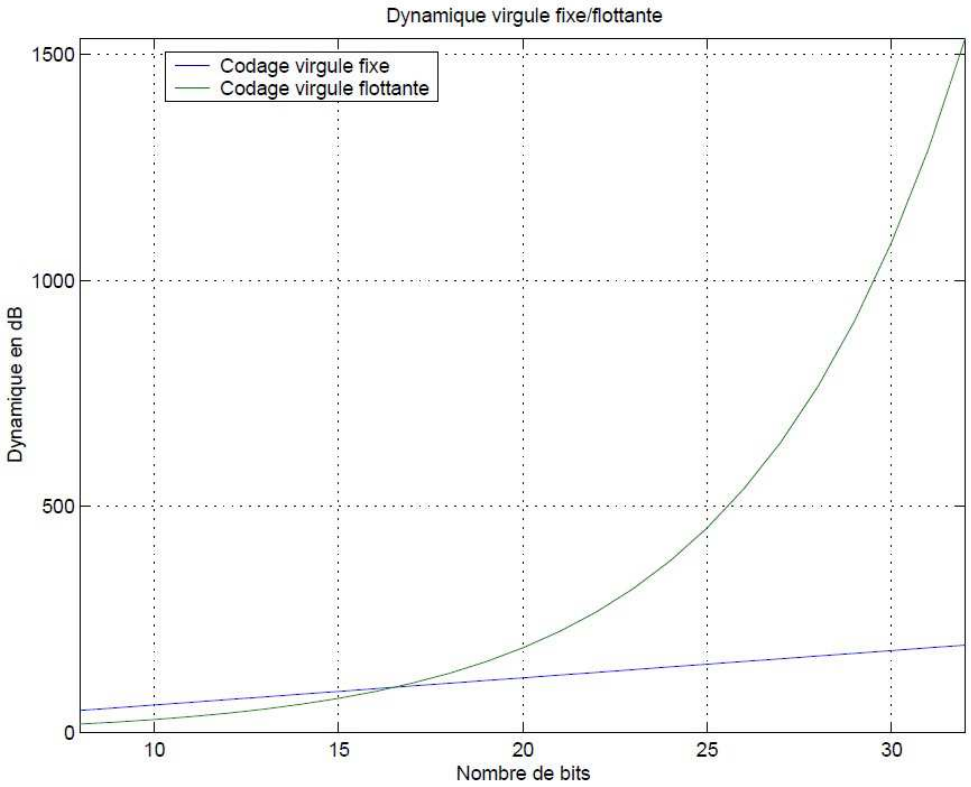

FIGURE 1.3 – Comparaison de l'évolution du niveau de la dynamique pour les représentations virgule fixe et virgule flottante

En comparant les équations (1.14) et (1.15), il est clair que la dynamique augmente exponentiellement pour le format virgule flottante et linéairement pour le format virgule fixe en fonction du nombre de bits accordés pour la représentation. Dans la figure 1.3, l'évolution de la dynamique en fonction du nombre de bits utilisés est présentée pour les deux formats virgule fixe et virgule flottante. Dans cet exemple, la taille de l'exposant représente le quart de la longueur totale du mot de code de la représentation. Nous pouvons observer que, lorsque la longueur d'un mot de code dépasse 16 bits, la dynamique du format virgule flottante devient plus large par rapport au format virgule fixe. Par conséquent, la représentation virgule flottante sur 32 bits peut être utilisée dans la plupart des applications sans aucun risque de débordement.

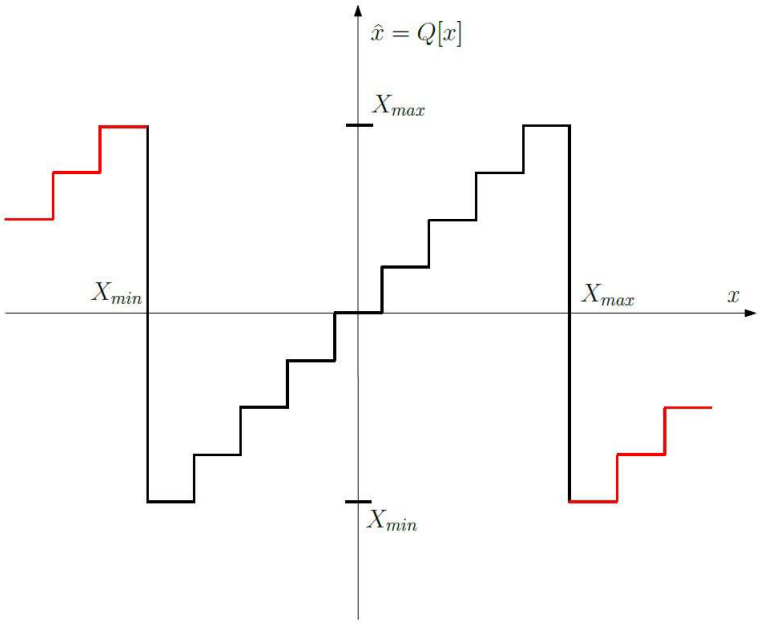

FIGURE 1.4 – Effet du débordement utilisant la technique de l'enveloppe autour de la valeur

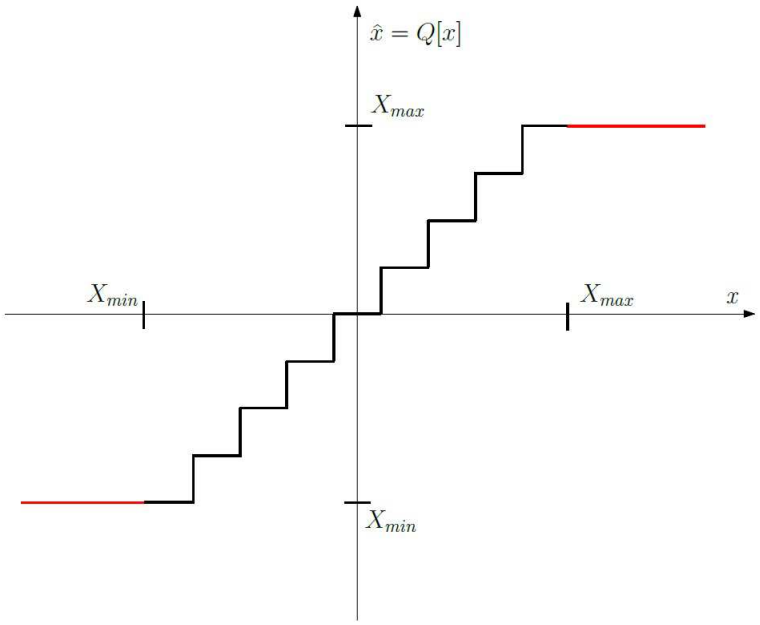

FIGURE 1.5 – Effet du débordement utilisant la technique de saturation

Le processus de codage d'un nombre en arithmétique virgule fixe correspond à une représentation d'une valeur réelle x par une autre valeur $\hat{x}$ appartenant au domaine de codage. A chaque fois que le nombre réel à coder dépasse l'intervalle des valeurs permises par la norme de codage (ce qui veut dire que $x \notin \left[\hat{X}_{min}, \hat{X}_{max}\right]$), un débordement se produit et une erreur est introduite. Il est alors nécessaire de définir une procédure de gestion de débordement afin d'être en mesure d'attribuer un mot de code dans cette situation. Deux techniques sont alors classiquement utilisées pour effectuer cette gestion du débordement. La première technique, décrite à la figure 1.4,

consiste à établir une enveloppe autour de la valeur du nombre réel à coder. Cette technique est équivalente à une arithmétique modulaire puisque toute valeur dépassant les bornes est remplacée par sa valeur modulo 2^b.

La deuxième méthode qui peut être appliquée est l'arithmétique de saturation. Dans ce cas, n'importe quelle valeur dépassant le domaine de codage est remplacée par le nombre représentable le plus proche (la borne maximale ou minimale). Cette technique est présentée par la figure 1.5. L'erreur introduite est plus petite que dans le cas de l'arithmétique modulaire. Cependant, à l'inverse de la technique de l'enveloppe autour de la valeur, l'implémentation de l'arithmétique de saturation nécessite une complexité matérielle plus importante et, par conséquent, son utilisation est limitée dans la pratique.

Comparaison du rapport Signal à Bruit de Quantification Pour les applications de traitement numérique de signal (DSP), la précision des calculs est mesurée le plus souvent à travers le critère de Rapport Signal à Bruit de Quantification (RSBQ). Le RSBQ est défini par le rapport entre la puissance du signal (P_x) et la puissance de l'erreur (P_e) due au bruit de quantification et il est généralement exprimé selon une échelle logarithmique :

$$RSBQ_{dB} = 10log\left(\frac{P_x}{P_e}\right) = 10log\frac{E\left[x^2\right]}{E\left[e^2\right]}. \tag{1.17}$$

Pour un signal x, sa puissance P_x peut s'exprimer en fonction de sa dynamique linéaire D_x et du rapport K_x entre la racine carrée de la puissance P_x et la dynamique D_x :

$$P_x = \left(K_x D_x\right)^2. \tag{1.18}$$

Par conséquent, dans le cas de l'arithmétique virgule fixe, le RSBQ suit une relation linéaire en fonction de la dynamique, selon :

$$\begin{aligned} RSBQ_{dB} &= 20log(D_x) + 20log(K_x) - 10log(P_e) \\ &= D_{dB} + 20log(K_x) - 10log(P_e) \end{aligned} \tag{1.19}$$

Comme le format virgule fixe possède une quantification uniforme, le RSBQ dépend linéairement de l'amplitude du signal. Lorsque l'amplitude du signal augmente, le rapport bruit de quantification devient plus large et le RSBQ s'améliore.

Considérons le cas d'un signal sinusoïdal à grande échelle avec une amplitude $A = 2^n$. La variance du signal est : $\sigma_x^2 = \frac{A^2}{2}$. Le RSBQ devient :

$$RSBQ = 20\log(2^n\sqrt{\frac{3}{2}}) \approx 1.76 + 6.02n \quad dB. \tag{1.20}$$

Pour conclure, dans le cas de la représentation virgule fixe avec un signal utilisant la totalité de la dynamique disponible, chaque bit additionnel conduit à l'augmentation du RSBQ de 6 dB approximativement.

Contrairement au cas de l'arithmétique virgule fixe, l'arithmétique virgule flottante possède l'avantage d'une étape de quantification proportionnelle à l'amplitude du signal. La valeur du RSBQ utilisant une échelle logarithmique est donnée par l'expression (1.21) et dépend du nombre de bits m utilisé pour la mantisse. En contre partie, le RSBQ de la représentation virgule flottante ne varie pas en fonction de l'amplitude du signal, il peut être considéré constant pour toutes les valeurs de x.

$$RSBQ = 10\log(\frac{E\left[x^2\right]}{e_{vf}}) = 10log(5.55 \times 2^{2m}) \approx 7.44 + 6.02m \tag{1.21}$$

Dans le tableau (1.2), une comparaison entre le RSBQ des deux formats virgule flottante et virgule fixe est présentée. Nous pouvons remarquer que, pour un même nombre de bits, l'arithmétique

virgule fixe garantit un RSBQ plus large que celui de l'arithmétique virgule flottante si le nombre est correctement mis à l'échelle.

Format	RSBQ (dB)
Précision simple	151
Précision double	326
virgule fixe 32 bits	194
virgule fixe 64 bits	387

TABLE 1.2 – Comparaison du RSBQ

Comparaison pour les opérations de calcul

D'après ce qui précède, il est clair que le format virgule flottante est plus performant que la représentation virgule fixe car il permet d'obtenir une dynamique suffisante et également une meilleure précision. Cependant, ce résultat est obtenu au prix d'une complexité des opérateurs arithmétiques. L'addition et la multiplication étant les deux opérations les plus génériques, nous étudions ci-après les coûts de réalisation de ces opérations.

L'addition Le nombre en format virgule fixe est nativement dans le format binaire. Sous cette hypothèse, il est possible d'additionner deux nombres représentés en virgule fixe en utilisant un circuit d'entraînement de report d'addition simple après l'alignement des points binaires de deux nombres. L'effort total pour l'addition en arithmétique virgule fixe ϵ^a_{fix} est égal à l'effort (le coût de réalisation) de l'addition binaire ϵ_a :

$$\epsilon^a_{fix} = \epsilon_a. \tag{1.22}$$

Cependant, le format virgule flottante est avant tout un format compressé. Il est donc nécessaire tout d'abord de décompresser le nombre en virgule flottante afin d'avoir accès au signe, à l'exposant et à la mantisse. Ensuite, l'exposant doit être normalisé en comparant les bits dans la partie exposant des deux nombres et en décalant les bits de la mantisse d'une manière telle que les exposants soient égaux. Si le bit de signe est réglé, les bits de la mantisse sont convertis à leurs formes en complément à deux avant l'addition. Après l'addition, le résultat doit être converti au format complément à deux s'il est négatif. Comparativement, l'addition de deux nombres en arithmétique virgule fixe est relativement plus simple que l'addition en format virgule flottante. Soit ϵ_{cmp} l'effort nécessaire pour la comparaison des exposants et soit ϵ_{cal} l'effort demandé pour le calcul des compléments à deux des deux nombres, l'effort total pour réaliser une addition en virgule flottante ϵ^a_{flt} :

$$\epsilon^a_{flt} = \epsilon_a + \epsilon_{cmp} + 2\epsilon_{cal}. \tag{1.23}$$

Les efforts ϵ_a, ϵ_{cmp} et ϵ_{cal} sont tous de l'ordre de $O(N)$ lorsque N est le nombre de bits accordés à la sortie de l'additionneur. Le temps pour la décompression du format virgule flottante est négligeable puisque cette opération est triviale du point de vue matériel.

La multiplication La multiplication de deux nombres binaires signés au format virgule fixe nécessite l'utilisation d'un multiplicateur matériel. Le résultat de la multiplication possède davantage de bits dans son niveau de dynamique et sa précision. La position du point binaire du nombre résultant en arithmétique virgule fixe est déterminée en prenant en compte les positions du point binaire des deux nombres impliqués dans la multiplication. Habituellement, les bits les moins significatifs du résultat sont rejetés par divers modes d'arrondis ou de troncature pour obtenir le résultat final. Par la suite, l'effort total de la multiplication en format virgule fixe ϵ^m_{fix} est ϵ_m c'est à dire le coût correspondant à une multiplication binaire, soit :

$$\epsilon^m_{fix} = \epsilon_m. \tag{1.24}$$

Dans le cas d'une multiplication entre des nombres représentés en virgule flottante, leurs exposants sont simplement additionnés, leurs mantisses et leurs bits de signe sont multipliés entre eux. L'effort total de la multiplication en virgule flottante ϵ^m_{flt} est :

$$\epsilon^m_{flt} = \epsilon_m + \epsilon_a. \tag{1.25}$$

La complexité algorithmique d'une multiplication (cas virgule flottante) peut être supérieure à $O(N^2)$ et le temps pour l'addition est $O(N)$ avec N est le nombre de bits accordés à la longueur du mot de code à la sortie du multiplicateur.

Comparaison de point de vue implémentation matérielle et logicielle

Chaque implémentation logicielle ou matérielle possède ses propres contraintes qui déterminent le choix de l'arithmétique la plus adaptée. Dans le cas d'une implémentation logicielle, les longueurs du mot de code sont fixées ou bien alors elles possèdent un nombre limité de représentations dans le format virgule flottante ou virgule fixe. Cependant, le format de la représentation en virgule fixe est flexible dans la mesure où le point binaire n'est pas fixe. Dans les plateformes logicielles typiques, les tailles d'un mot de code en arithmétique virgule fixe sont généralement d'un octet (8bits), moitié d'un mot (16 bits), un mot (32 bits) et aussi longue que mot double (64 bits). Les nombres en virgule flottante sont habituellement selon deux formats : soit la précision simple ou la précision double. De plus, il convient de préciser que les plateformes processeurs supportent les opérations des vecteurs de données SIMD (*Single Instruction Multiple Data*) [35]. De telles instructions fonctionnent sur un ensemble de données de même taille et de même type rassemblées en un bloc de données, d'une taille fixe, appelé vecteur. Il est alors possible de configurer le chemin de données selon une taille normalisée (le nombre de bits est généralement multiple de quatre ou huit) afin de contrôler plus précisément les longueurs des mots de code en arithmétique virgule fixe.

L'implémentation matérielle ouvre également des horizons pour des unités en virgule flottante personnalisées [25] [27]. Dans [40], les spécifications des opérateurs en virgule flottante sont décrites en utilisant une bibliothèque C++ paramétrée d'opérateurs en virgule flottante. Il en résulte une génération automatique des implémentations optimales des opérateurs en virgule flottante dans le matériel de telle sorte que le temps de calcul est suffisamment petit pour atteindre la fréquence désirée de l'opération. L'impact du nombre de bits accordés à l'opérateur en virgule flottante sur le niveau de la dynamique et la précision de l'opération n'est pas aussi simple que dans le cas de système des nombres en virgule fixe. En définitive, il s'avère difficile de faire un choix entre le format virgule fixe et le format virgule flottante sans explorer explicitement toutes les opérations.

Certaines stratégies [15] fournissent une bibliothèque paramétrable permettant de faire un choix entre les deux formats à partir de l'étude de différents compromis obtenus à partir de simulations. Dans [11] et [13], le niveau de la dynamique est considéré pour optimiser le format d'un nombre, i.e. minimiser la dynamique des calculs tout en contrôlant la précision à un niveau permettant de ne pas dépasser les coûts de la virgule flottante. Nous pouvons également remarquer que de nombreuses applications de traitement du signal sont linéaires et ces systèmes peuvent faire usage du facteur d'échelle et donc travailler avec une gamme dynamique normalisée. En outre, les applications de traitement du signal et en particulier les algorithmes de communications numériques sont conçus pour fonctionner en présence de bruit de canal dont la valeur est beaucoup plus grande par rapport au bruit de quantification introduit, même lorsqu'une quantification 16 bits est utilisée. Par conséquent, la précision requise pour la mise en œuvre de ces algorithmes n'est pas très stricte. Dans de telles circonstances, il est possible de profiter pleinement de la représentation à virgule fixe. Par conséquent, les applications de traitement numérique de signal et de communications numériques sont le domaine d'intérêt de cette thèse.

Au lieu d'utiliser un arbre d'addition à virgule fixe, d'autres travaux [26] utilisent un système de représentation hybride où les nombres à additionner sont normalisés relativement au plus

grand exposant et où l'addition à virgule fixe est utilisée pour ajouter la mantisse. Cela réduit l'effort de calcul de l'exposant à chaque fois.

Une autre possibilité consiste à utiliser le format virgule fixe double. Dans ce cas, un seul bit est utilisé pour représenter l'exposant et la mantisse qui correspond à un nombre signé en virgule fixe. Une telle technique permet de réaliser un compromis entre dynamique et précision en étendant le concept de l'arithmétique virgule flottante. Une troisième alternative utilise un système de nombres en virgule flottante en bloc [24]. Dans ce cas, lorsqu'un groupe de nombres partagent un même exposant, la mantisse de ces nombres peut être différente. Cette technique a été implémentée pour des nombres en virgule flottante dans plusieurs algorithmes de traitement de signal [57] [73] dont notamment la bibliothèque *DirectX* pour les applications graphiques à haut niveau de dynamique. L'idée commune à toutes ces approches hybrides vise à tirer avantage de la grande plage de dynamique et la haute précision de la représentation en virgule flottante tout en conservant la flexibilité de la représentation en virgule fixe.

Conclusion

L'arithmétique virgule flottante présente l'avantage d'une dynamique et d'une précision supérieures à l'arithmétique virgule fixe. A l'opposé, l'arithmétique virgule fixe, plus flexible, possède des opérateurs moins complexes et conduit à des implémentations moins coûteuses en termes de consommation d'énergie et de ressources matérielles. Cependant, l'implantation en virgule fixe nécessite la maîtrise de la précision des calculs qui est évaluée par des méthodes de simulation ou par des méthodes analytiques que nous présenterons dans la section 1.3. Les méthodes analytiques nécessitent la modélisation des bruits de quantification, c'est pourquoi nous abordons ces notions dans la section suivante.

1.2 Modélisation du bruit de quantification

La conversion de la représentation virgule flottante au format virgule fixe conduit à l'élimination de certains bits lors des calculs, ce qui engendre l'apparition de bruits de quantification. Dans cette section, la modélisation du bruit de quantification est évoquée.

1.2.1 Processus de quantification

Le mécanisme d'assigner une séquence de chiffres binaires pour une représentation d'une valeur réelle (analogique) x est réalisé par le processus de quantification. L'opération de quantification est présentée dans (1.26), où la valeur du signal x est transformée en une représentation à virgule fixe notée par $\hat{x}$:

$$x \rightarrow \hat{x} = Q(x). \tag{1.26}$$

Il s'agit d'une procédure non linéaire qui génère une perte de précision puisque seulement un nombre fini de valeurs possibles peuvent être représentées. Plus précisément, lorsque b bits sont utilisés pour représenter un nombre en virgule fixe, 2^b valeurs distinctes peuvent être représentées. L'erreur résultante de la différence entre la valeur réelle x et sa représentation en virgule fixe $\hat{x}$ est appelée bruit de quantification :

$$e(x) = x - \hat{x}. \tag{1.27}$$

La résolution de la représentation, notée q, est donnée par la différence minimale entre deux valeurs consécutives représentables. Sa valeur est déterminée par la position du bit de poids le plus faible (LSB) (2^{-n}). Deux modes de quantification sont particulièrement utilisés : la quantification par arrondi et celle par troncature.

La quantification par arrondi

Lorsque la quantification par arrondi est appliquée (figure 1.6), l'amplitude du signal est arrondie au niveau de quantification le plus proche. L'erreur maximale introduite est $\pm\frac{1}{2}LSB$. Par conséquent, l'erreur de quantification introduite ($e(x)$) dans ce cas est bornée dans l'intervalle $[-\frac{q}{2}, \frac{q}{2}]$, soit :

$$\hat{x} = Q(x) = \Delta_i + \frac{q}{2}, \quad \forall x \in]\Delta_i, \Delta_{i+1}] \,. \tag{1.28}$$

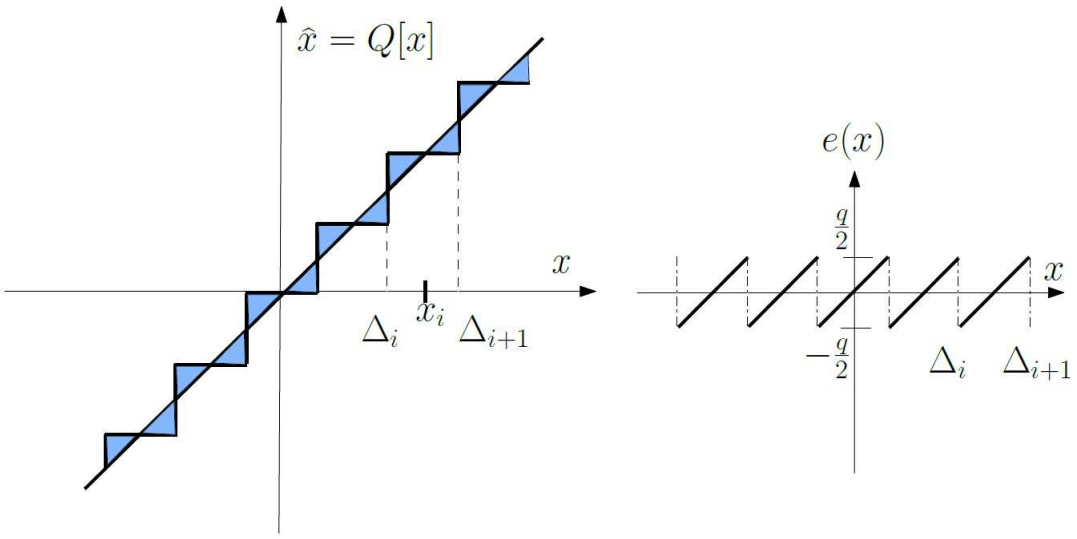

FIGURE 1.6 – Processus de quantification par arrondi

La quantification par troncature

La méthode de quantification par troncature (figure 1.7) consiste à choisir le niveau de quantification inférieur pour une représentation du signal. Par conséquent, l'erreur de quantification est toujours positive ($e(x) \in]0, q]$) et un décalage est introduit. On a donc :

$$\hat{x} = \Delta_i, \quad \forall x \in]\Delta_i, \Delta_{i+1}] \,. \tag{1.29}$$

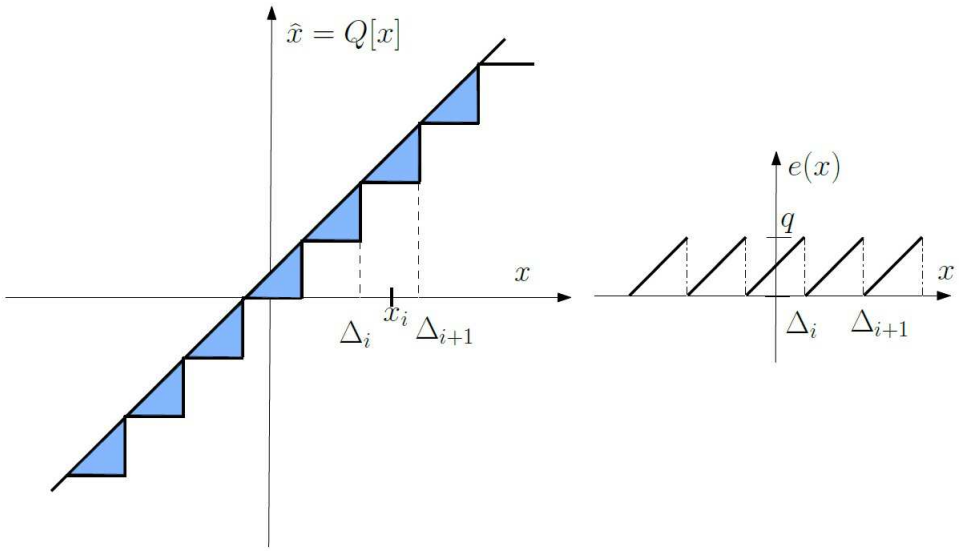

FIGURE 1.7 – Processus de quantification par troncature

Supposons que le signal x à quantifier soit un signal aléatoire de densité de probabilité (FDP) continue $p_x(x)$. La quantification de x engendre la génération d'un nouveau signal $\hat{x}$ de densité de probabilité $p_{\hat{x}}(\hat{x})$ discrète qui est constituée de M valeurs notées p_i. Chaque p_i est la probabilité que le signal x appartient à un intervalle I_i. Par conséquent, les bornes de l'intervalle I_i sont liées au mode de quantification utilisé (par arrondi ou par troncature). Les valeurs p_i sont déterminées par :

$$p_i = \int_{I_i} p_x(x)dx. \tag{1.30}$$

Les valeurs discrètes p_i sont séparées par le pas de quantification q. Par conséquent la fonction densité de probabilité (FDP) du signal $\hat{x}$ est :

$$p_{\hat{x}}(\hat{x}) = \sum_{i=1}^{M} p_i \delta(\hat{x} - iq). \tag{1.31}$$

Dans [93], Widrow a montré que lorsque la fonction caractéristique du signal d'entrée x, i.e. la transformée de Fourier de $p_x(x)$, est nulle alors le signal $\hat{x}$ peut être modélisé par les deux variables aléatoires (x, e) de densités de probabilité $p_x(x)$ et $p_e(e)$. Par conséquent, le processus de quantification peut être modélisé par l'introduction d'un bruit additif. La sortie du quantificateur est alors égale à la somme du signal d'entrée x et de l'erreur de quantification comme représenté sur la figure 1.8.

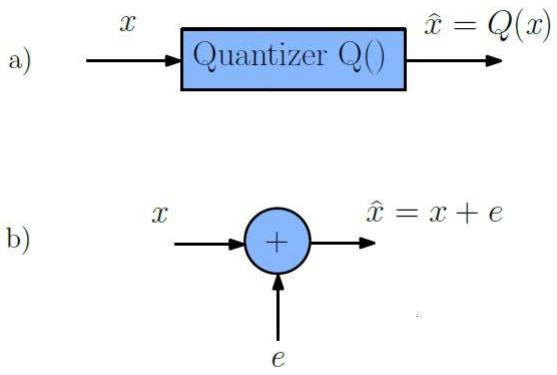

FIGURE 1.8 – Modélisation de bruit de quantification additif

Nous étudierons, dans ce qui suit, plus en détail le bruit de quantification e en différenciant le cas d'un signal d'entrée à amplitude discrète ou à amplitude continue.

1.2.2 Signal à amplitude continue

Soit x un signal à amplitude continue avec une FDP continue. Les différents intervalles I_i sont déterminés selon le mode de quantification choisi (arrondi ou troncature).

Cas du mode quantification par troncature

Dans ce cas, l'intervalle I_i est l'intervalle entre deux valeurs représentables par le codage :

$$I_i = [x_i, x_{i+1}[. \tag{1.32}$$

Comme vu précédemment, $e(x) \in [0, q[$. Sa FDP est rectangulaire (figure 1.9), elle est donnée par l'équation :

$$p_e(e) = \frac{1}{q}\text{rect}(\frac{e - \frac{q}{2}}{q}). \tag{1.33}$$

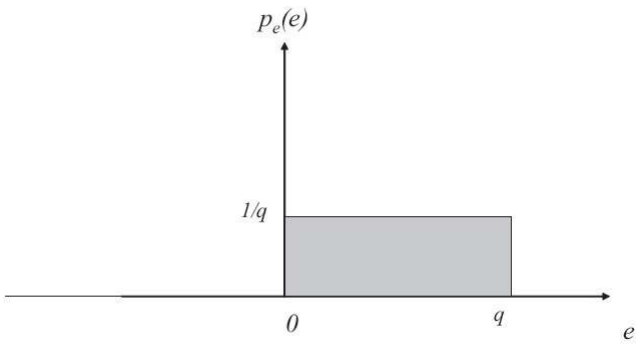

FIGURE 1.9 – Densité de probabilité de la loi de troncature continue

Dans ce cas, la moyenne, la variance et la puissance du bruit de quantification e sont données par les équations suivantes :

$$\mu_e = \int_{-\infty}^{\infty} e\ p_E(e)\ de = \int_0^q \frac{1}{q}\ e\ de = \frac{q}{2}. \tag{1.34}$$

$$\sigma_e^2 = \int_{-\infty}^{\infty} (e - \mu_e)^2 \; p_E(e) de = \int_0^q \frac{1}{q} \Big(e - \frac{q}{2}\Big)^2 \; de = \frac{q^2}{12}. \tag{1.35}$$

$$E(e^2) = \mu_e^2 + \sigma_e^2 = \frac{q^2}{3}. \tag{1.36}$$

Cas du mode quantification par arrondi

Dans ce mode de quantification, l'intervalle I_i et la FDP du bruit de quantification e (figure 1.10) sont définis par :

$$I_i =]p_i - \frac{q}{2}, p_i + \frac{q}{2}[\tag{1.37}$$

$$p_e(e) = \frac{1}{q} \text{rect}(\frac{e}{q}) \tag{1.38}$$

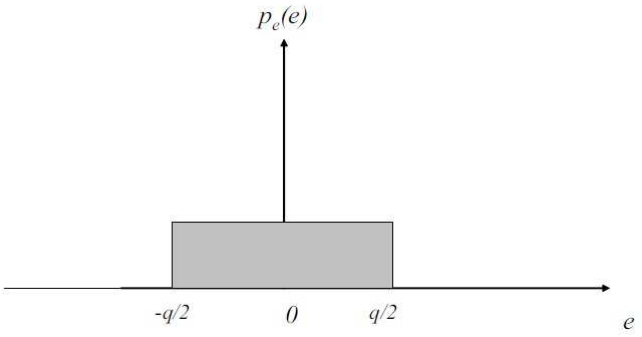

FIGURE 1.10 – Densité de probabilité de la loi d'arrondi continu

La variance et la moyenne de e s'expriment selon :

$$\mu_e = \int_{-\infty}^{\infty} e \; p(e) de = \int_{-q/2}^{q/2} \frac{1}{q} \; e \; de = 0 \tag{1.39}$$

$$\sigma_e^2 = \int_{-\infty}^{\infty} (e - \mu_e)^2 \; p(e) de = \int_{-q/2}^{q/2} \frac{e^2}{q} de = \frac{q^2}{12}. \tag{1.40}$$

Sa puissance est alors seulement égale à sa variance de valeur $\frac{q^2}{12}$.

Statistiques du bruit de quantification

Auto-corrélation du bruit de quantification e Dans cette section, une étude de la corrélation entre deux bruits est présentée. Nous considérons deux variables aléatoires e_1 et e_2. La FDP conjointe des deux bruits est définie dans [96] par :

$$p_{e_1,e_2}(e_1, e_2) = rect(\frac{e_1}{q}) rect(\frac{e_2}{q}) [p_{x_1,x_2}(x_1, x_2) * (f(e_1).f(e_2))] \tag{1.41}$$

où * désigne le produit de convolution. Dans [96], il a été montré que la FDP conjointe s'écrit :

$$p_{e_1,e_2}(e_1, e_2) = \frac{1}{q} rect(\frac{e_1}{q}) \frac{1}{q} rect(\frac{e_2}{q}) = p_{e_1}(e_1) p_{e_2}(e_2). \tag{1.42}$$

Par conséquent, la FDP conjointe est décomposée en un produit des FDP des deux bruits indépendants. De plus, en supposant les processus aléatoires stationnaires, la fonction d'auto-corrélation $\mathcal{R}_{ee}(\theta)$ à un instant θ est définie par l'expression suivante :

$$\mathcal{R}_{ee}(\theta) = \sigma_e^2 \delta(\theta) + \mu_e^2. \tag{1.43}$$

avec δ distribution de Dirac. L'erreur de quantification est donc un bruit blanc. Pour une valeur nulle de θ, la puissance est retrouvée.

Corrélation avec le signal Le moment d'ordre 2 du signal quantifié $\hat{x}$ est étudié. Puisque $\hat{x} = x + e$, le moment d'ordre 2 est :

$$E(\hat{x}^2) = E(x^2) + E(e^2) + 2E(xe). \tag{1.44}$$

D'autre part, le moment d'ordre 2 est la dérivée seconde de la fonction caractéristique à l'origine. Par conséquent la dérivée seconde de la fonction caractéristique de $\hat{x}$ donne une nouvelle expression de la corrélation entre x et e, suivant :

$$E(xe) = \sum_{i=-\infty}^{\infty} q\Phi(-i\frac{2\pi}{q})\frac{(-1)^i}{\pi i}. \tag{1.45}$$

Donc si $\Phi(-i\frac{2\pi}{q}) = 0$ quelle que soit la valeur de i, l'erreur de quantification n'est pas corrélée au signal. Cette condition est vérifiée dès que le pas de quantification q est inférieur à l'écart-type du signal [96]. Dans le cas contraire, le bruit généré a une puissance supérieure à celle du signal et le bon fonctionnement du système n'est plus assuré.

Conclusion

Pour conclure, nous pouvons retenir que le bruit généré par la quantification d'un signal à amplitude continue est un bruit blanc, additif, non corrélé avec le signal et indépendant des autres signaux. De plus, sa distribution est uniforme sur son espace de représentation.

1.2.3 Signal à amplitude discrète

Nous nous intéressons dans cette section à l'étude d'un signal à amplitude discrète dont la quantification est effectuée par élimination de plusieurs bits d'une donnée. Cette conversion conduit à la génération d'un bruit à amplitude discrète. Les caractéristiques de ce bruit diffèrent selon le mode de quantification considéré.

Loi de quantification par troncature L'implantation de cette loi est très simple puisque seulement les bits les plus significatifs sont conservés. La figure 1.11 représente la distribution du bruit où k représente le nombre de bits éliminés.Un modèle fondé sur une équiprobabilité entre les valeurs 0 et 1 est présenté dans [31]. La moyenne et la variance de e sont données par les équations suivantes :

$$\mu_e = \frac{q}{2}(1 - 2^{-k}) \tag{1.46}$$

$$\sigma_e^2 = \frac{q^2}{12}(1 - 2^{-2k}). \tag{1.47}$$

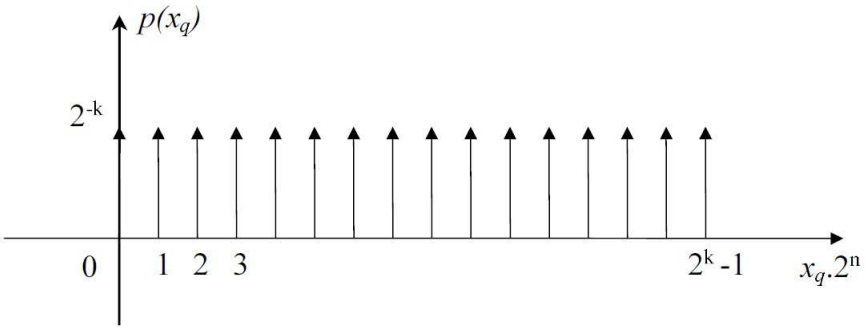

FIGURE 1.11 – Densité de probabilité de la loi de troncature discrète

Loi de quantification par arrondi Dans le cas où l'amplitude de signal est discrète, le domaine de représentation n'est pas centré sur 0. Par conséquent, le problème consiste à déterminer la valeur à affecter à une donnée sur la médiane d'un intervalle de représentation $p_i + \frac{q}{2}$. Cette valeur est fixée à la valeur supérieure représentable p_{i+1}. Par conséquent, ce choix conduit à une moyenne non nulle de l'erreur et un biais plus faible que dans le mode de quantification par troncature. Le signal quantifié $\hat{x}$ est donné par l'expression suivante :

$$y = Q(x) = \begin{cases} p_l & \text{si } p_l \leq x < p_l + \frac{q}{2} \\ p_{l+1} & \text{si } p_l + \frac{q}{2} < x \leq p_{l+1} \\ p_{l+1} & \text{si } x = p_l + \frac{q}{2}. \end{cases} \tag{1.48}$$

La moyenne de la loi par arrondi n'est plus nulle comme représenté sur la figure 1.12.

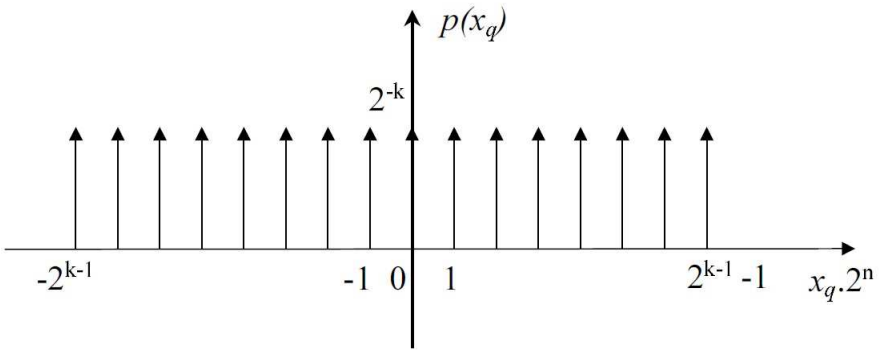

FIGURE 1.12 – Densité de probabilité de la loi d'arrondi discrète

La variance et la moyenne sont données par les expressions suivantes [31] :

$$\mu_e = \frac{q}{2}(2^{-k}), \tag{1.49}$$

$$\sigma_e^2 = \frac{q^2}{12}(1 - 2^{-2k}). \tag{1.50}$$

Loi de quantification par arrondi convergent La loi d'arrondi convergent a été introduite dans [65] afin d'éliminer complètement le biais présent au niveau de l'erreur. La seule différence de cette loi par rapport à la loi d'arrondi conventionnel est le traitement de la valeur médiane dont la valeur alterne entre la valeur supérieure et inférieure de la représentation. Par conséquent, le biais devient nul et la valeur du signal quantifié est :

$$y = Q(x) = \begin{cases} p_l & \text{si } p_l \leq x < p_l + \frac{q}{2} \\ p_{l+1} & \text{si } p_l + \frac{q}{2} < x \leq p_{l+1} \\ p_l \text{ ou } p_{l+1} & \text{si } x = p_l + \frac{q}{2} \end{cases} \tag{1.51}$$

La distribution du bruit est présentée par la figure 1.13. Sa moyenne est nulle et sa variance est donnée par l'équation suivante :

$$\sigma_e^2 = \frac{q^2}{12}(1 + 2^{-2k+1}). \tag{1.52}$$

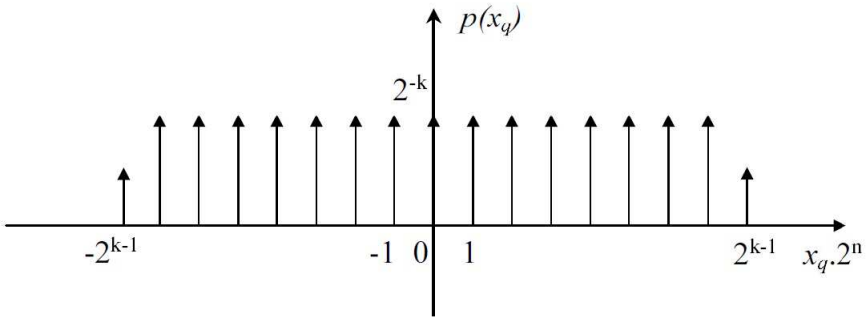

FIGURE 1.13 – Densité de probabilité de la loi d'arrondi convergente discrète

Conclusion

La distribution du bruit de quantification généré par la quantification d'un signal à amplitude discrète n'est plus continue mais discrète. Le nombre de bits éliminés k intervient dans la moyenne et la variance de ce bruit.

1.2.4 Conclusion

La quantification d'un signal à amplitude continue ou discrète conduit à la génération d'un bruit de quantification. Ce bruit est blanc, additif, non corrélé avec le signal et indépendant des autres bruits. La moyenne et la variance de ce bruit sont liées au mode de quantification utilisé (arrondi ou troncature) et au type de signal. Ces valeurs sont résumées dans le tableau 1.3, où l_1 est le nombre de bits de poids faibles nuls.

Paramètres statistiques	Amplitude-continue	Amplitude discrète	
		$z = x \times y$, $z = x + y$	$z = x \times C$
Arrondi convergent			
μ_e	0	0	0
σ_e^2	$\frac{q^2}{12}$	$\frac{q^2}{12}(1 - 2^{-2k+1})$	$\frac{q^2}{12}(1 - 2^{-2(k-l_1)+1})$
Arrondi			
μ_e	0	$\frac{q}{2}(2^{-k})$	$\frac{q}{2}(2^{-(k-l_1)})$
σ_e^2	$\frac{q^2}{12}$	$\frac{q^2}{12}(1 - 2^{-2k})$	$\frac{q^2}{12}(1 - 2^{-2(k-l_1)})$
Troncature			
μ_e	$\frac{q}{2}$	$\frac{q}{2}(1 - 2^{-k})$	$\frac{q}{2}(1 - 2^{-(k-l_1)})$
σ_e^2	$\frac{q^2}{12}$	$\frac{q^2}{12}(1 - 2^{-2k})$	$\frac{q^2}{12}(1 - 2^{-2(k-l_1)})$

TABLE 1.3 – Paramètres statistiques du bruit généré

1.3 Évaluation de la précision

L'utilisation de l'arithmétique à virgule fixe engendre une dégradation de la précision des calculs. La partie fractionnaire du mot de code représenté en virgule fixe est ajustée en évaluant les effets du bruit de quantification en se basant sur un compromis entre la précision demandée et le coût du circuit. Ce compromis fait l'objet de cette thèse puisque nous visons dans les chapitres suivants à évaluer la précision de plusieurs applications de traitement numérique de signal et de communication numérique afin de déterminer un format en virgule fixe approprié assurant un bon compromis. La perte en précision est généralement déterminée en calculant une métrique qui peut être obtenue à travers la comparaison de la sortie d'une implémentation en virgule fixe avec la sortie d'une implémentation référence qui est généralement une implémentation en virgule flottante double précision. L'erreur due aux calculs utilisant le format virgule flottante en double précision est très petite. Par conséquent, l'implémentation double précision est généralement très bonne et considérée comme une implémentation avec une arithmétique de précision infinie. Lors de la conception d'un système, un seuil minimal est prédéfini pour la métrique afin de valider l'implémentation en virgule fixe par rapport à celle de référence. Pour déterminer la métrique de l'évaluation de la précision en sortie de l'application, deux approches sont utilisées. La première approche consiste à évaluer cette métrique par simulation de l'algorithme en virgule fixe et en virgule flottante puis en comparant les deux sorties des deux simulations. Cependant, cette approche nécessite un temps important de simulation pour explorer l'espace de conception en virgule fixe. Pour cette raison, une seconde approche est proposée pour remédier à ce problème majeur de la simulation. Cette approche consiste évaluer analytiquement la métrique de précision en propageant un modèle de bruit au sein du graphe flot de l'algorithme. De plus, pour une application complexe qui comporte des opérateurs non linéaires en terme de bruit de quantification, il peut être intéressant d'opter pour une méthode [75] basée sur une combinaison des deux approches analytique et simulation pour constituer ce qu'on appelle une approche hybride. Nous commençons dans cette section par présenter les métriques de précision les plus utilisées, ensuite nous présenterons les approches d'évaluation par simulation, les approches analytiques et

l'approche hybride.

1.3.1 Les métriques de précision

Plusieurs métriques permettent de quantifier l'erreur obtenue en utilisant un système en virgule fixe en les comparant avec la/les sorties de référence. Pour les systèmes de communications numériques, la mesure du taux d'erreur binaire (TEB) constitue l'une des métriques les plus utilisées pour évaluer leurs performances. La différence entre les TEB obtenus avec la référence et la spécification en virgule fixe peut être considérée pour analyser l'effet de la quantification sur la sortie du système. Plus généralement, le rapport RSBQ entre la puissance du signal y et la puissance du bruit de quantification b vue précédemment est fréquemment utilisé pour mesurer l'impact des systèmes en virgule fixe. Cette métrique exprimée en pratique en décibel (dB) est très utilisée pour les applications de traitement numérique de signal.

$$RSBQ_{dB} = 10log\left(\frac{P_y}{P_b}\right) \tag{1.53}$$

D'autres métriques sont également envisageables. Une première solution consiste à définir comme métrique l'intervalle $\mathcal{D} = [b_{min}, b_{max}]$ définissant les bornes maximale et minimale du bruit de quantification. Il est alors possible de garantir que cet intervalle soit borné. Une solution alternative, présentée dans [28], vise à mesurer le nombre de bits n'ayant pas été affectés par le bruit de quantification ce qui est équivalent à évaluer les bits représentant significativement le signal utile. Il est également possible de s'intéresser à la valeur maximale de l'erreur de quantification. Ainsi, dans [62], la notion de quantification maximale est proposée comme métrique pour évaluer l'impact de l'arithmétique virgule fixe. De même, pour des opérations de filtrage numérique, la métrique considérée dans [42] correspond à l'erreur sur la réponse impulsionnelle introduite par la quantification des coefficients du filtre. L'utilisation des métriques de précision semble être simple, cependant, il existe plusieurs inconvénients. Dans le cas des systèmes de communications numériques, il est nécessaire de transmettre un grand nombre d'échantillons afin de garantir que la dégradation du TEB ne soit pas un phénomène aléatoire mais qu'elle est causée par l'utilisation de l'arithmétique virgule fixe. Ceci augmente le temps de simulation qui devient prohibitif surtout lorsqu'il s'agit de réaliser l'optimisation de la longueur du mot de code qui nécessite de répéter les simulations en virgule fixe.

Une autre voie pour évaluer la précision d'une représentation en virgule fixe consiste [77] à considérer l'erreur quadratique moyenne (EQM) entre le signal quantifié et la référence soit, de manière équivalente, à considérer la puissance totale du signal d'erreur entre le signal en précision infinie et sa représentation en virgule fixe [43]. L'EQM peut être obtenue par simulation ou alors sous forme analytique comme une fonction du mot de code en virgule fixe.

1.3.2 Les méthodes d'évaluation de la précision par simulation

Dans cette partie, nous présentons les techniques permettant d'évaluer la métrique de précision par simulation. Afin de bien étudier cette stratégie, nous commençons par évoquer les techniques adéquates pour la simulation d'un système en virgule fixe. Pour garantir un résultat précis, il faut utiliser un nombre d'échantillons en entrée du système (N_{ech}) large. Nous considérons N_{ops} opérations présentes dans l'algorithme et N_i le nombre de fois que la métrique de précision doit être évaluée. Le nombre de points N_{pts} à calculer exprimé par $N_{pts} = N_{ech}N_{ops}N_i$ dévoile le rôle important du choix d'un simulateur capable d'aboutir à des temps raisonnables de simulations.

Le temps de simulation donné par :

$$T_{sim} = \tau N_{pts} = \tau N_{ech}N_{ops}N_i \tag{1.54}$$

est une fonction du nombre de points N_{pts} et du temps de simulation d'un point τ. En effet, à chaque changement du format des données, une nouvelle simulation est nécessaire pour optimiser la spécification en virgule fixe.

Plusieurs travaux ont été menés afin de réduire le temps de simulation. Les différentes stratégies proposées visent (i) à minimiser le temps de calculs τ par la conception d'un simulateur efficace, (ii) à limiter le nombre d'échantillons N_e, ou bien encore (iii) à réduire le nombre N_i d'itérations comme envisagé dans [16],[17].

Simulation virgule fixe

Nous présentons dans cette partie différentes méthodes statistiques fondées sur un outil de simulation en virgule fixe. Cet outil émule les mécanismes de la représentation virgule fixe en termes de dépassement et de quantification sur un poste de travail utilisant l'arithmétique virgule flottante. Pour effectuer l'émulation de ces mécanismes, les méthodes existantes utilisent les propriétés de la surcharge des opérateurs. Cette dernière technique, présentée par Kim [61], permet une spécification virgule fixe à travers le type *gFix* [59] [60]. Lors d'une assignation, une conversion du format entre la donnée de droite et celle de gauche est réalisée en suivant les règles spécifiées au sein d'attributs. La spécification en entrée du simulateur correspond au programme source écrit en C++, où le type des données en virgule flottante est remplacé par le type *gFix*.

Dans ce cas, la surcharge des opérateurs conduit à un temps d'exécution de l'algorithme plus long qu'une simulation virgule flottante. Pour avoir une idée sur les temps d'exécution, nous pouvons citer l'exemple du temps de simulation de la librairie SystemC qui est en moyenne 540 fois supérieur à celui obtenu en virgule flottante [33].

Dans [61], un nouveau type *pFix* a été proposé pour réduire le temps de simulation de la méthode précédente. Cette méthode est basée sur l'optimisation de l'utilisation des chemins des données de l'unité de calculs en virgule flottante. Par conséquent, ce type enregistre les données en virgule fixe en utilisant la mantisse des données en virgule flottante ce qui réduit le temps de simulation. Dans le cas d'un filtre IIR d'ordre 4, le temps de simulation est supérieur de 7,5 fois à celui de la virgule flottante. Une des limitations du type *pfix* réside dans la taille maximale de la mantisse dont le nombre de bits accordés est 53. Dans [18], une technique similaire est utilisée pour l'accélération des bibliothèques en virgule fixe du SystemC.

Pour réduire le temps de simulation, une autre technique, proposée par Coster [34], conduit à un codage efficace des données de l'algorithme en exploitant tous les types d'entiers disponibles sur la machine.

La performance du type de données en virgule fixe peut être également accélérée en utilisant une architecture convenable du processeur (comme dans le cas du DSP). Ainsi, les techniques proposées dans [64][48], conçues initialement pour la génération automatique du code en virgule fixe, peuvent être réutilisées pour accélérer l'exécution en ciblant une affectation du format de données en virgule fixe sur les plateformes DSP. Dans cet esprit, l'outil *System Generator de Xilinx* [94] fournit un ensemble de modules personnalisés en virgule fixe pouvant être utilisés sous Matlab-Simulink, et permettant un prototypage rapide sur le matériel d'algorithmes de traitement du signal.

Évaluation de la puissance de bruit

En utilisant la simulation virgule fixe d'un système sous contrainte, il est possible de déterminer la puissance du bruit de quantification en utilisant la technique de la figure 1.14.

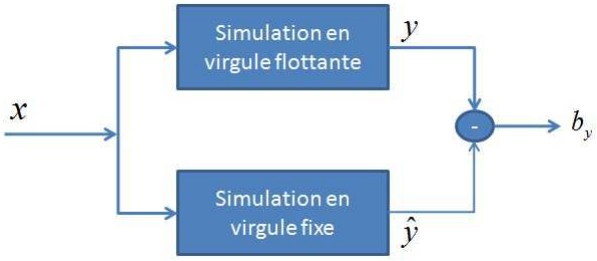

FIGURE 1.14 – Mesure de la puissance de bruit de quantification causé par la virgule fixe

Le système est implémenté en utilisant *un design* de référence avec une très haute précision, i.e. la perte de précision liée à la simulation en virgule flottante peut être considérée comme négligeable par rapport à une implémentation en précision infinie. Finalement, en sortie de la simulation du système en format à virgule fixe, le signal de sortie $\hat{y}$ permet de mesurer le bruit de quantification à travers la différence entre les deux sorties des deux simulations en virgule fixe et en virgule flottante. Afin de réduire le temps de simulation, il est possible de réduire le coût des simulations en virgule fixe. Comme proposé dans [68], Ceci peut se faire en analysant la qualité de la puissance de bruit en fonction du nombre d'échantillons N_{ech}.

Mesure de nombre de bits significatifs

Une autre stratégie d'évaluation présentée dans [28] est la méthode basée sur l'arithmétique stochastique [29] qui modifie aléatoirement le dernier bit de chaque donnée. Le système est simulé N fois en utilisant les mêmes données à l'entrée de l'application. Pour chaque simulation, N_i correspond à un résultat en sortie R_i. Une estimation de la valeur en sortie $\overline{R}$ est donnée à travers la moyenne des échantillons R_i :

$$\overline{R} = \frac{\sum_{i=1}^{N} R_i}{N}. \tag{1.55}$$

Dans ce cas, le nombre de bits communs entre la valeur réelle R et la valeur estimée $\overline{R}$ correspond au nombre de bits significatifs. La variable aléatoire $N(\frac{\overline{R}-R}{s})$ suit une loi de Student à $N-1$ degrés de liberté, où s est défini par :

$$s = \sqrt{\frac{\sum_{i=1}^{N}(R_i - \overline{R})^2}{N-1}}. \tag{1.56}$$

Le nombre de bits significatifs est déterminé par la relation suivante :

$$C_{\overline{R}} = log_2\left(\frac{\sqrt{N}|\overline{R}|}{s\tau_\beta}\right). \tag{1.57}$$

De plus, la variable τ_β du test de Student pour $N-1$ degrés de liberté et de probabilité β permet de valider cette méthode.Cette méthode permet de diminuer le nombre requis de simulations par un facteur 3, cependant ces performances sont limitées par des contraintes sur la nature du bruit de quantification. En effet, elle considère que le bruit en sortie du système suit une loi gaussienne centrée, ce qui n'est plus valide avec le mode de quantification par troncature présenté dans la deuxième section. De plus, cette méthode ne fournit aucune indication sur l'évolution moyenne au cours du temps de l'application en virgule fixe.

Les bornes de l'erreur

La métrique de l'intervalle de l'erreur de quantification en sortie du système est utilisée dans [74], en se basant sur la Théorie des Valeurs Extrêmes (TVE). Cette théorie considère que les valeurs extrêmes d'une suite d'échantillons suivent la distribution de Gumbel donnée par :

$$G(x) = e^{-e^{\left(\frac{x-\mu}{\sigma}\right)}}. \tag{1.58}$$

où $\sigma = \frac{s\sqrt{6}}{\pi}$ et $\mu = \overline{x} - \sigma\lambda$. Les termes s et $\overline{x}$ représentent respectivement l'écart-type et la moyenne des échantillons, et $\lambda = 0.5772$ est la constante d'Euler.

La stratégie de cette méthode est fondée sur la simulation de N échantillons auxquels une valeur en sortie x_i est affectée. Par conséquent, l'écart-type et la moyenne des échantillons sont donnés par les équations suivantes :

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i, \tag{1.59}$$

$$s = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \overline{x})^2}{N}}. \tag{1.60}$$

La valeur maximale obtenue dépend de la probabilité de dépassement fixée au départ. Elle est déterminée par :

$$a = \mu - \sigma ln\left(ln(\frac{1}{1-f})\right). \tag{1.61}$$

Cette valeur a est la valeur maximale de la sortie du système avec une probabilité de dépassement f. Cette méthode possède l'avantage de diminuer de manière importante le nombre des simulations puisqu'elle permet de donner la valeur maximale de l'injection des bruits en entrée.

Conclusion

Les approches pour évaluer la précision sont universelles et peuvent être appliquées sans aucune restriction sur n'importe quel type de système. Plusieurs approches basées sur la simulation en virgule fixe ont été proposées dans la littérature. Cependant, le temps pris pour la simulation et la génération des vecteurs tests représentatifs des scénarios réels est un défi. Néanmoins, l'intérêt de réaliser des simulations en virgule fixe devient évident lorsqu'il n'y a aucune alternative pour estimer la dégradation de la précision. Des méthodes analytiques ont été proposées pour évaluer cette précision afin de réduire significativement le temps d'optimisation. Ces approches font l'objet du prochain paragraphe.

1.3.3 Les approches analytiques de l'évaluation de la précision

Afin de s'affranchir des temps d'exécution prohibitifs des approches pour évaluer la précision par simulation, des approches analytiques ont été proposées. Ces méthodes sont basées sur l'évaluation des métriques de précision présentées précédemment par le biais de modèles mathématiques permettant d'analyser le bruit en sortie du système. Comme montré sur la figure 1.15, l'objectif est de déterminer analytiquement le bruit de quantification b_y, ou ses caractéristiques, et son influence sur le système de telle sorte que la sortie $\hat{y}_{an}$ soit statistiquement équivalente à la sortie $\hat{y}$ obtenue par la simulation en virgule fixe. Cette technique vise à calculer analytiquement les propriétés statistiques de l'erreur de quantification e_y et, ensuite, à modéliser la composante additive b_y pour que $\hat{y}_{an}$ soit statistiquement équivalente à $\hat{y}$.

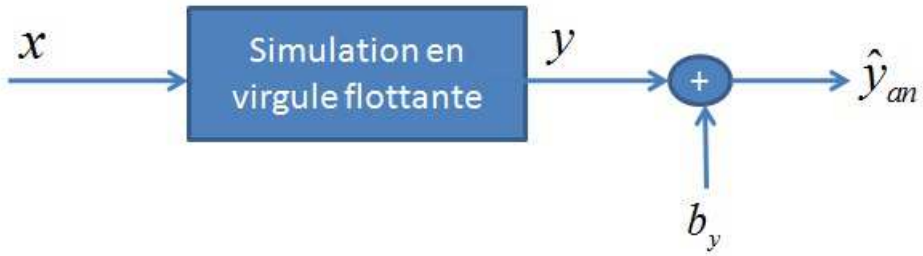

FIGURE 1.15 – Estimation analytique de l'erreur causée par les opérations en virgule fixe

Certaines techniques analytiques sont le résultat de l'application de deux théories :
- Le modèle analytique du bruit de quantification connu par le modèle *pseudo bruit de quantification* (PQN :Pseudo-Quantification-Noise). Ce modèle est basé sur l'étude des statistiques de l'erreur de quantification en sortie du quantificateur virgule fixe.
- Le modèle de propagation de bruit basé sur l'application de la théorie de perturbation. Le modèle linéaire de propagation du bruit est utilisé afin de déterminer l'influence sur les calculs de la représentation des entrées en précision finie .

Nous présentons dans cette section différentes approches analytiques pour évaluer la précision en virgule fixe en se basant essentiellement sur deux métriques : la puissance de bruit et les bornes de l'erreur.

Le modèle PQN

Ce modèle stochastique, encore nommé modèle de bruit de quantification de Widrow [92], définit un processus aléatoire statistiquement équivalent au phénomène de la quantification uniforme. En d'autres termes, en utilisant le modèle PQN, l'effet de la quantification uniforme peut être modélisé avec un bruit additif b_x pour obtenir un signal $\hat{x}_{an}$ qui est statistiquement équivalent au signal $\hat{x}$ obtenu après la quantification uniforme du signal x. Cela est illustré par la figure 1.16. De plus, le modèle PQN impose des conditions telle qu'une équivalence statistique (même propriété statistique) entre le signal bruit b_x et l'erreur e_x sous lesquelles les signaux $\hat{x}$ et $\hat{x}_{an}$ peuvent être établis.

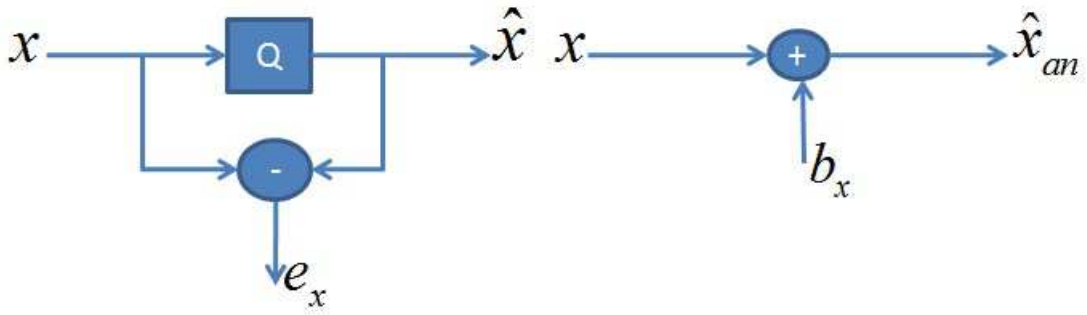

FIGURE 1.16 – Modèle analytique du bruit de quantification

L'intérêt majeur du modèle PQN repose sur le fait qu'il permet d'estimer les statistiques du signal e_x. La première caractéristique utilisée par ce modèle concerne la fonction de densité de probabilité (FDP).

En effet, le signal erreur b_x est uniformément distribué, sa densité spectrale de puissance est blanche, et sa puissance est déterminée par la valeur du pas de quantification ainsi que

par celle de sa valeur moyenne qui dépend du mode de quantification (arrondi ou troncature). Deuxièmement, le bruit additif n'est pas statistiquement corrélé avec le signal quantifié. Par conséquent, les statistiques du signal quantifié peuvent être obtenues en ajoutant simplement les statistiques du pseudo bruit de quantification b_x aux statistiques du signal original. Ces propriétés statistiques sont vérifiées lorsque le pas de quantification est faible comparé à la dynamique du signal. Plus précisément, il est nécessaire que la fonction caractéristique de la FDP du signal soit suffisamment petite afin d'éviter le repliement des fonctions caractéristiques du signal quantifié correspondant. Dans une telle situation, le processus de quantification est appelé quantification lisse *(smooth-quantization)* afin de souligner le fait [92] qu'un faible changement dans la valeur du signal quantifié n'entraîne qu'une variation minime de la sortie de du système. L'étude des applications composées d'opérateurs non lisses *(un-smooth operators)* constitue l'objet principal de ce travail de thèse, en particulier la recherche de modèles analytiques permettant d'évaluer leur précision. Ces travaux seront présentés au sein du prochain chapitre.

Modèle analytique pour les bornes de l'erreur

Arithmétique d'intervalles (AI) Cette approche évalue analytiquement les bornes de l'erreur de quantification en se basant sur l'arithmétique d'intervalles. Son principe général se base sur la propagation des intervalles des données en entrées selon des règles définies pour aboutir à un encadrement de la sortie. Soit x et y des données en entrée pouvant prendre des valeurs respectivement dans l'intervalle $\mathcal{D}_x = [x_{min}, x_{max}]$ et $\mathcal{D}_y = [y_{min}, y_{max}]$. Lors de la propagation de ces intervalles dans un système composé d'opérations telles que l'addition, la multiplication et la multiplication par une constante, les intervalles en sortie sont donnés par les équations suivantes :

$$\begin{aligned} \mathcal{D}_{x+y} &= [x_{min} + y_{min}, x_{max} + y_{max}] \\ \mathcal{D}_{xy} &= [min(x_{min}y_{min}, x_{min}y_{max}, x_{max}y_{min}, x_{max}y_{max}) \\ &, \quad max(x_{min}y_{min}, x_{min}y_{max}, x_{max}y_{min}, x_{max}y_{max})] \\ \mathcal{D}_{ax} &= [ax_{min}, ax_{max}] \text{ si } a > 0 \\ &= [ax_{max}, ax_{min}] \text{ si } a < 0. \end{aligned} \tag{1.62}$$

Cette technique comporte une limitation importante liée à l'estimation pessimiste faite de l'intervalle de définition des données qui devient parfois très large car le résultat fourni est celui du pire cas puisque cette arithmétique ne prend pas en compte les corrélations entre les données. De plus, cette méthode ne s'applique que lorsque la sortie ne dépend que des valeurs d'entrée.

Arithmétiques Affines (AA) L'arithmétique affine a été proposée dans [36] afin de s'affranchir des inconvénients de l'arithmétique d'intervalles, et particulièrement de la non prise en considération de la corrélation entre les données. Soit x une donnée décomposée de la façon suivante :

$$x = x_0 + \sum_{i=1}^{n} \epsilon_i x_i \quad avec \quad \epsilon_i \in [-1, 1]. \tag{1.63}$$

Les symboles ϵ_i sont indépendants et représentent les coordonnées de x sur les composantes x_i qui sont les bornes de x. Par conséquent, x_0 est la moyenne des valeurs possibles prises par x. Le qualificatif affine vient du fait du développement d'une forme affine en fonction des symboles ϵ_i, lors de la propagation des intervalles de définition dans les opérations. L'intérêt de cette méthode est la prise en compte de la corrélation entre les différentes données. Soient x et y deux données décomposées de la façon suivante :

$$x = x_0 + \sum_{i=1}^{n} \epsilon_i x_i, \tag{1.64}$$

$$y = y_0 + \sum_{i=1}^{n} \epsilon_i y_i. \tag{1.65}$$

Pour la somme de ces deux données, c'est une fonction affine des symboles ϵ_i :

$$x + y = x_0 + y_0 + \sum_{i=1}^{n} \epsilon_i (x_i + y_i). \tag{1.66}$$

En ce qui concerne l'opération de la multiplication, l'opération xy n'est plus une fonction affine car elle s'écrit selon :

$$xy = x_0 y_0 + \sum_{i=1}^{n} (x_0 y_i + y_0 x_i)\epsilon_i + \sum_{i=1}^{n} x_i \epsilon_i \sum_{j=1}^{n} y_j \epsilon_j. \tag{1.67}$$

Afin de rendre l'opération sous une forme affine, une majoration du dernier terme de l'équation précédente est introduite par l'introduction de z_{n+1} tel que :

$$z_{n+1} > \sum_{i=1}^{n} |x_i| \sum_{j=1}^{n} |y_j|. \tag{1.68}$$

Par conséquent la multiplication de x par y introduit le symbole ϵ_{n+1} qui est le facteur associé à z_{n+1} et s'écrit sous cette forme :

$$x + y = x_0 y_0 + \sum_{i=1}^{n} (x_0 y_i + y_0 x_i)\epsilon_i + z_{n+1}\epsilon_{n+1}. \tag{1.69}$$

Cette arithmétique fournit un intervalle plus fin que celui donné par l'arithmétique d'intervalles puisqu'elle utilise les corrélations spatiales entre les données. Cependant, cette technique produit parfois un intervalle trop lâche car elle ne tient pas compte des corrélations temporelles. Elle peut donc être limitée pour des applications de traitement du signal où les données sont temporellement corrélées (par exemple, les filtres FIR).

Méthode basée sur la théorie de la perturbation Wadekar propose dans [91] une approche analytique pour évaluer les bornes de l'erreur de quantification basée sur un développement de Taylor d'ordre 2 de l'erreur en sortie. Soit un opérateur à deux entrées u et v et δ_u et δ_v les erreurs au pire cas sur ces données. La fonction de sortie de cet opérateur est $f(u, v)$. L'erreur en sortie s'écrit comme suit :

$$\delta_f = \frac{\partial f}{\partial u}(\delta_u) + \frac{\partial f}{\partial v}(\delta_v) + \frac{1}{2!}\left[\frac{\partial^2 f}{\partial u^2}(\delta_u)^2 + 2\frac{\partial^2 f}{\partial u \partial v}(\delta_u \delta_v) + \frac{\partial^2 f}{\partial v^2}(\delta_v)^2\right]. \tag{1.70}$$

Cette approche, basée sur la propagation des bruits au sein du système, s'applique aussi bien sur des systèmes LTI que non-LTI. Cependant, elle ne peut être appliquée dans le cas de systèmes récursifs ainsi que celui des systèmes itératifs composés d'opérateurs non lisses et non linéaires en termes de bruit.

Approches analytique de la puissance de bruit

Le bruit engendré par la quantification d'un signal est donné par le modèle PQN. Les contributions du bruit causé par une quantification grossière des signaux déjà quantifiés suivent aussi les dynamiques du théorème de quantification. La puissance de quantification est calculée comme une fonction des moments de premier ordre et du second ordre du signal d'erreur de quantification. L'amplitude du bruit de quantification injecté dans le système par différentes opérations en arithmétique virgule fixe est déterminée selon le format virgule fixe choisi ainsi que le mode de quantification. Si le format virgule fixe est suffisamment large, aucun bruit de quantification est injecté.

Propagation du bruit et théorie de la perturbation Le changement infinitésimal, créé en sortie du système en raison des calculs en arithmétique virgule fixe, n'affecte pas le comportement macroscopique total du système. L'application de la théorie de perturbation est essentiellement une alternative pour estimer approximativement l'impact de la quantification sous certaines circonstances. Les erreurs causées par le calcul avec des opérateurs en arithmétique virgule fixe sont des petites perturbations dont l'effet sur la sortie peut être maîtrisé avec un bon degré de précision à l'aide de techniques d'approximation linéaire. En effet, cette approximation introduit une tolérance envers les erreurs de quantification pour une application et pose la question de la petite taille du pas de quantification qui devrait être choisi pour être classifiée comme lisse (smooth). Mention contraire, on peut supposer que les quantificateurs sont lisses. En outre, l'opérateur lui-même peut être considéré comme lisse ou non lisse en fonction de son comportement fonctionnel. Une opération est considérée comme lisse si la sortie est une fonction continue et différentiable de ses entrées. La technique de propagation du bruit basée sur la théorie des perturbations, tel que présenté dans la présente section, est applicable seulement aux systèmes comprenant des opérateurs lisses et quantificateurs lisses.

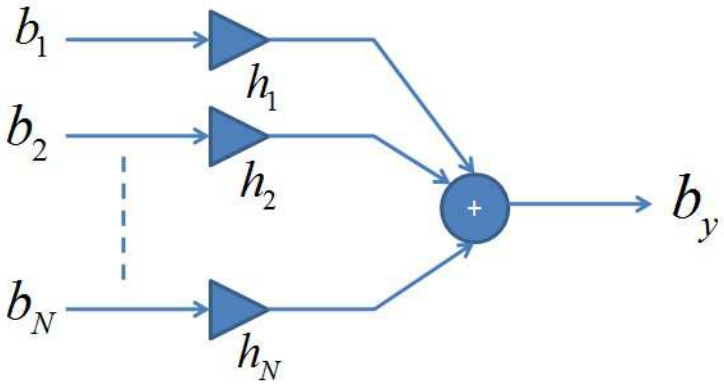

FIGURE 1.17 – Propagation du bruit de quantification vers la sortie du système

Considérons un opérateur binaire dont les entrées sont x et y et la sortie est z. Si les signaux d'entrée sont perturbés par les bruits b_x et b_y pour obtenir respectivement $\hat{x}$ et $\hat{y}$, la sortie est perturbée par b_z pour obtenir $\hat{z}$. En d'autres termes, tant que l'opérateur en virgule fixe est lisse, l'impact de petites perturbations à l'entrée se traduit par une perturbation en sortie de l'opérateur sans aucune modification de son comportement macroscopique. Selon la théorie des perturbations, le bruit en sortie b_z est une combinaison linéaire des bruits en entrée b_x et b_y :

$$b_z = v_1 b_x + v_2 b_y. \tag{1.71}$$

Dans ce cas, l'hypothèse sous-jacente est que les termes de bruit b_x et b_y ne soient pas corrélés entre eux, ni avec les signaux d'entrée. Cette hypothèse découle de l'hypothèse suivante relative au modèle PQN de génération du bruit de quantification : le bruit de quantification est indépendant du signal. Les termes v_1 et v_2 sont obtenus à partir de l'approximation de Taylor du premier ordre de la fonction continue et différentiable f :

$$\overline{z} = f(\overline{x}, \overline{y}) \simeq f(x, y) + \frac{\partial f}{\partial x}(x, y).(\overline{x} - x) + \frac{\partial f}{\partial y}(x, y).(\overline{y} - y). \tag{1.72}$$

Par conséquent, les termes v_1 et v_2 sont donnés par :

$$v_1 = \frac{\partial f}{\partial x}(x, y) \qquad v_2 = \frac{\partial f}{\partial y}(x, y). \tag{1.73}$$

Les termes v_1 et v_2 peuvent varier dans le temps. De plus, cette méthode n'est pas limitée uniquement aux opérations mais elle peut être appliquée au niveau fonctionnel sur tous les opé-

rateurs appartenant à un même chemin de données et ainsi propager le bruit de quantification de la source jusqu'à la sortie.

Soit y, le signal en sortie du système, le bruit correspondant b_y causé par la quantification est obtenu par la somme de la contribution individuelle des sources de bruit dans le système comme montré dans la figure 1.17. Si $h_i(n)$ représente la réponse impulsionnelle du chemin entre la $i^{ème}$ source de bruit et la sortie du système, la contribution de la source de bruit b_y est calculée par la somme de produits de convolution de la source de bruit et de la réponse impulsionnelle. Considérons un système comportant, à son entrée, N sources de bruit $b_i, i = 1, ..., N$. Il existe alors N chemins indépendants entre chaque source de bruit présente en entrée et la sortie du système. La contribution de la source de bruit b_i peut alors être calculée en prenant en compte la réponse impulsionnelle $h_i(k), k = 0, 1, ...$ du chemin entre la $i^{ème}$ source de bruit et la sortie du système. Le bruit global en sortie b_y s'exprime alors comme la somme des N produits de convolution individuels. Dans le cas où les réponses impulsionnelles varient dans le temps, il est donné par :

$$b_y = \sum_{i=1}^{N} \sum_{k=-\infty}^{\infty} h_i(k, n) b_i(n - k) \tag{1.74}$$

où $h_i(k, n)$ représente le $k^{ième}$ coefficient de la réponse impulsionnelle mesurée à l'instant n. La puissance du bruit en sortie est obtenue comme le moment du second ordre du bruit à la sortie du système et elle est donnée par :

$$\begin{aligned} \mathrm{E}[b_y^2] &= \mathrm{E}([\sum_{i=1}^{N_e} \sum_{k=-\infty}^{\infty} h_i(k, n) b_i(n - k))^2] \\ &= \sum_{i=1}^{N_e} \sum_{k=-\infty}^{\infty} \mathrm{E}[h_i^2(k, n)] \mathrm{E}[b_i(n - k)^2] \\ &+ \sum_{i=1}^{N_e} \sum_{j=1, j\neq i}^{N_e} \sum_{k=-\infty}^{\infty} \sum_{l=-\infty, l\neq k}^{\infty} \mathrm{E}[h_i(k, n) h_j(l, n)] \mathrm{E}[b_i(n - k) b_j(n - l)]. \end{aligned} \tag{1.75}$$

L'expression précédente peut se mettre sous la forme d'une somme des paramètres statistiques de la source du bruit comme suit :

$$\mathrm{E}[b_y^2] = \sum_{i=1}^{N_e} K_i \sigma_{b_i}^2 + \sum_{i=1}^{N_e} \sum_{j=1, j\neq i}^{N_e} L_{ij} \mu_{b_i} \mu_{b_j} \tag{1.76}$$

où μ_{b_i} et σ_{b_i} désignent respectivement la valeur moyenne et la variance du bruit présent à l'entrée numéro i. Les termes K_i et L_{ij} sont constants et dépendent de la réponse impulsionnelle h_i associée au chemin n°i. Cette approche supporte les systèmes récursifs et non récursifs dans le cas des systèmes LTI (Linear-Time Invariant) et non-LTI. L'expression de ces coefficients est :

$$K_i = \sum_{k=-\infty}^{\infty} \mathrm{E}[h_i^2(k)], \tag{1.77}$$

$$L_{ij} = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \mathrm{E}[h_i(k) h_j(l)]. \tag{1.78}$$

Afin de déterminer les différentes réponses impulsionnelles caractérisant les relations entrées/sortie, le graphe flot de données est partitionné en plusieurs sous-graphes acycliques à des fins d'analyse. En utilisant le modèle de propagation, les fonctions des chemins de ces sous graphes sont déterminées dans le domaine de la transformée en Z. La fonction individuelle de chemin de chaque source de bruit jusqu'à la sortie est obtenue à l'aide de la combinaison des fonctions de ces sous-graphes par substitution de variable.

Une autre technique basée sur l'arithmétique affine est proposée dans [54][44] respectivement pour les systèmes LTI et non-LTI. Cette technique consiste à définir une forme affine pour chaque

source de bruit dans le système en prenant en compte sa valeur moyenne et sa variance. Profitant de l'avantage du modèle de propagation linéaire, les formes affines de bruit sont propagées à travers le système par simulations. Les valeurs de K_i et L_{ij} sont obtenues par l'estimation des formes affines à partir de l'expression du bruit en sortie. Dans le cas d'un systèmes récursif, cette opération est répétée plusieurs fois jusqu'à obtenir convergence de la forme affine caractérisant le bruit en sortie.

Dans [85], les coefficients de l'expression de la puissance de bruit en sortie dans l'équation (1.76) sont obtenus à partir du développement au second ordre de la différence entre les expressions du bruit en arithmétique virgule fixe et en arithmétique virgule flottante. Dans ce cas, les coefficients de Taylor sont directement liés aux coefficients L_i et K_{ij} présentés précédemment. Ils sont obtenus en résolvant un système d'équations linéaires en effectuant le minimum de simulations en virgule fixe ainsi qu'une seule simulation en virgule flottante.

1.3.4 Autres effets de la quantification

Nous avons jusqu'à présent concentré notre analyse sur l'erreur de quantification engendrée par la représentation en arithmétique virgule fixe des données présentes en entrée d'un opérateur. Cependant, l'opération de quantification en arithmétique virgule fixe peut également modifier le comportement global d'un système notamment lorsque l'opération de quantification porte sur les paramètres décrivant ce système. Certains changements peuvent ainsi être particulièrement importants puisqu'ils peuvent affecter la stabilité du système dans certains cas. Ces effets sont étudiés essentiellement selon deux catégories.

La première catégorie est l'impact de la quantification des paramètres du système. Dans [55], l'effet de la quantification des coefficients sur les bornes de l'intervalle du signal en sortie est déterminé.

A partir des outils d'analyse de l'arithmétique d'intervalles et de l'arithmétique affine, il est ainsi démontré que, dans certains cas, la quantification des coefficients d'un système peut engendrer, en sortie du système, un bruit dont la puissance est plus importante que celle obtenue lors de la quantification des données présentes à l'entrée du système. L'effet de la quantification d'un coefficient sur la sensibilité d'un système global est également étudié dans le domaine de la transformée en z et celui de la transformée de Fourier par les auteurs de [45] et [87]. Enfin, signalons également l'approche proposée dans [51] permettant d'évaluer cette sensibilité à partir d'une technique relativement simple basée sur la simulation.

L'effet non linéaire engendré par le processus de quantification peut également jouer un rôle fondamental dans le cas où le système comporte des boucles récursives. Cette situation constitue le cadre principal de ce travail de thèse, et plus particulièrement celui obtenu lorsque ces boucles sont constituées d'opérateurs non lisses tels que les opérateurs de décision. Ainsi, sous certaines conditions, des oscillations peuvent apparaître en sortie du système [71], [86]. Il est alors important de pouvoir détecter l'apparition de cycles limites engendrés par l'opération de quantification. D'autres travaux , e.g. [20], [21], et [90], précisent les conditions garantissant la stabilité asymptotique d'un système. Ces études sont essentiellement basées sur des approches par simulations en raison de la difficulté mathématique rencontrée pour obtenir des expressions analytiques. Aussi, afin de limiter le temps requis pour réaliser ces simulations de façon exhaustive [56], d'autres travaux [53] ont permis d'accélérer l'étape de simulation en utilisant conjointement des techniques d'analyse basées sur l'arithmétique affine.

1.3.5 Bilan des deux approches

En premier lieu, les techniques statistiques par simulations en virgule fixe sont particulièrement intéressantes en raison de leur simplicité de réalisation mais également puisqu'elles s'appliquent directement à n'importe quel type de système (linéaire, non linéaire, bouclé, ...). Malheureusement, afin d'obtenir des résultats statistiquement significatifs, il est nécessaire de réaliser ces simulations sur un nombre important de données en entrée (Monte Carlo) ce qui conduit rapi-

dement à des temps de simulation prohibitifs, et plus particulièrement lorsqu'il est nécessaire de balayer l'espace complet des formats en virgule fixe. L'autre approche est l'approche d'évaluation basée sur les modèles analytiques pour évaluer mathématiquement une métrique de précision. Cette approche est basée sur l'évaluation du comportement de bruit par le biais d'une expression mathématique dont le calcul n'est effectué qu'une seule fois. Une fois cette expression analytique disponible, les paramètres statistiques décrivant le bruit s'obtiennent directement et en une seule fois. Par conséquent le temps nécessaire pour l'évaluation de la précision est relativement limité puisqu'il correspond au coût de l'évaluation de cette expression analytique. Ce temps est donc nettement plus faible que celui requis par les approches basées sur la simulation en virgule fixe (cf. figure 1.18). Cependant, l'obtention d'une expression analytique décrivant les propriétés statistiques du bruit de quantification peut se révéler difficile pour certains types de systèmes notamment ceux possédant des opérateurs non lisses.

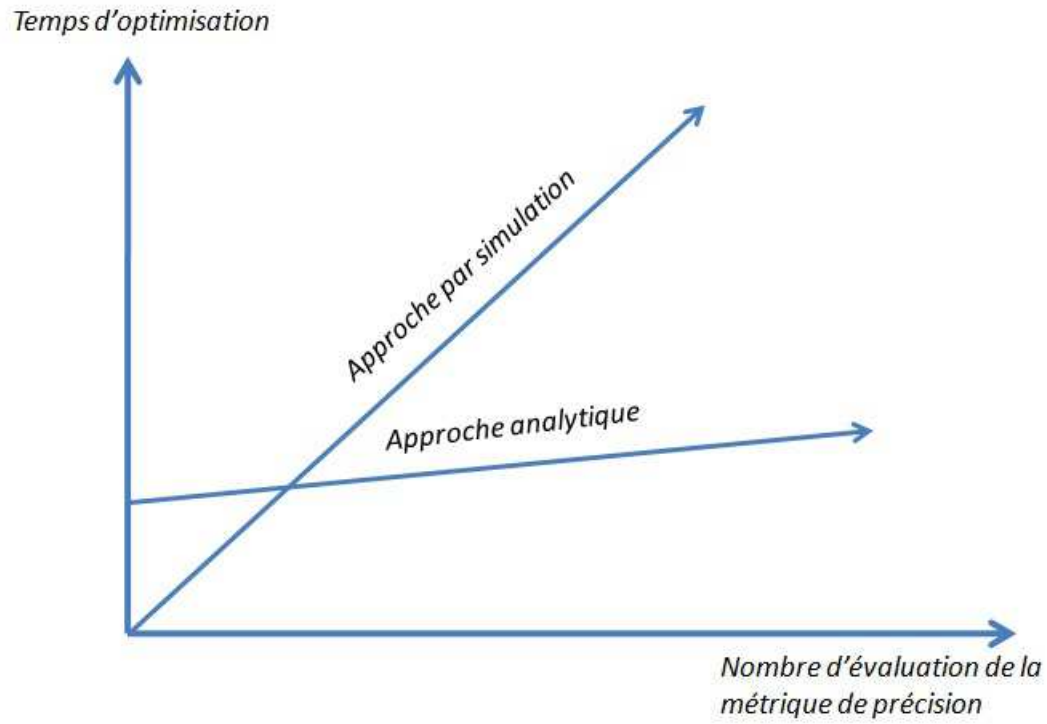

FIGURE 1.18 – Les temps d'optimisation des deux approches en fonction de nombre d'évaluation

1.3.6 Les approches hybrides

Afin d'accélérer les temps d'optimisation, des approches hybrides basées conjointement sur des modèles analytiques et sur des simulations ont été proposées [75]. Ces approches consistent à utiliser des modèles analytiques afin de décrire les éléments du système comportant des opérateurs élémentaires tels que additions, multiplications,..., mais également à utiliser la simulation pour évaluer le comportement d'opérateurs complexes tels que les opérateurs non lisses (*un-smooth*). Une étape d'évaluation sélective partitionne tout d'abord l'ensemble du système en plusieurs grappes d'opérateurs lisses séparées par des opérateurs non lisses. Pour chaque grappe d'opérateurs lisses, un modèle de sources de bruits est dérivé afin d'analyser l'effet des opérations de quantification, puis ces valeurs sont propagées à travers le système jusqu'à ce qu'un opérateur non lisse soit rencontré. Dans cette approche hybride, les grappes d'opérateurs lisses sont évaluées par le biais d'une approche analytique et les opérateurs non lisses sont évalués par le biais de simulations. Une telle approche permet d'utiliser pleinement les modèles analytiques pour estimer le comportement des blocs lisses, soit donc de bénéficier au maximum de leurs avantages.

1.4 Conclusion

Nous avons vu dans ce chapitre les différentes types de codage de données et en particulier celui relatif à l'arithmétique en virgule fixe. Cette arithmétique possède d'immenses avantages pratiques essentiellement pour les applications de traitement numérique de signal. Cependant, elle conduit à la génération de bruit de quantification qui se propage au sein de l'application et modifient la précision des calculs. Ainsi, l'évaluation de la précision en virgule fixe est une étape primordiale dans la conception des systèmes en virgule fixe. Afin d'évaluer cette précision, deux approches fondamentales ont été présentées : l'approche par simulation et l'approche analytique. Cette dernière est basée sur l'évaluation d'une métrique de précision par le biais d'une expression mathématique ce qui permet de réduire le temps d'optimisation par rapport à celui requis par l'approche par simulation qui devient prohibitif lorsqu'il s'agit de balayer l'espace des formats en virgule fixe. Les techniques d'évaluation de la précision par les modèles analytiques restent cependant limitées à des systèmes comportant des opérateurs lisses. Par conséquent, il est inévitable d'utiliser les techniques basées sur la simulation pour certaines applications (ou parties d'application) comportant des opérateurs non lisses (*un-smooth operators*). Une approche hybride basée sur la combinaison des deux approches pour les applications comportant des opérateurs non lisses a été proposée dans [75]. Toujours dans l'optique de l'amélioration des temps d'optimisation, nous proposons dans le prochain chapitre, une approche purement analytique d'évaluation de la précision pour les opérateurs non lisses qui sera appliquée à l'évaluation de la précision en virgule fixe des algorithmes de décodage sphérique tel que le SSFE (Selective Spanning with Fast Enumeration) composé d'opérateurs lisses et d'opérateurs non lisses (opérateurs de décision).

Chapitre 2

Évaluation analytique de la précision des algorithmes de décodage sphérique

Sommaire

Par rapport à la précision infinie, l'utilisation d'opérations en virgule fixe introduit un bruit de quantification qui modifie la valeur des nombres. Les erreurs de quantification dont les caractéristiques ne peuvent pas être déterminées par le modèle PQN sont classées comme des erreurs non lisses (un-smooth). En effet, ce modèle fonctionne correctement tant que le pas de quantification reste faible devant la dynamique du signal comme dans le cas des opérations en arithmétique virgule fixe. Dans le cas des opérateurs *non lisses*, les erreurs de quantification sont très grandes et souvent comparables à la dynamique du signal quantifié. Dans ce cas, les hypothèses de validité du modèle PQN ne sont plus vérifiées et donc le signal quantifié sera éloigné de la valeur vraie.

Un modèle analytique visant à prédire les statistiques de l'erreur à la sortie d'un opérateur *non lisse* nécessite la connaissance du signal en entrée et les caractéristiques du bruit de quantification associé. De plus, en présence de plusieurs opérateurs *non lisses*, les caractéristiques de l'erreur de quantification ne sont pas uniquement corrélées avec le signal, mais aussi avec les autres erreurs issues des opérateurs *non lisses* précédents. Il convient de préciser que dans le cas des

opérateurs *non lisses*, il n'existe pas de modèle analytique permettant d'évaluer leurs précisions en arithmétique virgule fixe.

L'objectif de ce chapitre est de proposer un modèle analytique pour évaluer la précision de l'algorithme de décodage sphérique SSFE. Nous commençons par proposer, dans la première section, une étude des différents types d'opérateurs. Nous présentons dans la seconde partie un modèle analytique pour évaluer la précision d'un opérateur *non lisse* correspondant à l'opérateur de décision en se basant sur la probabilité d'erreur de décision causée par la quantification. La troisième partie du chapitre est consacrée à l'extension de cette méthode au cas d'une cascade d'opérateurs de décision. Enfin, le modèle analytique proposé sera appliqué, dans la dernière partie du chapitre, à l'algorithme de décodage sphérique SSFE.

2.1 Les opérateurs lisses et non lisses

Dans cette section, nous présentons les différents types d'opérateurs présents dans les systèmes numériques en faisant la distinction entre opérateurs *lisses* et *non lisses*.

2.1.1 Introduction

Généralement, les signaux présents en entrées d'un système numérique peuvent être considérés comme des processus aléatoires à temps discret [76], [84]. En revanche, les opérateurs présents dans un système peuvent souvent être traités comme des opérations déterministes qui produisent des processus aléatoires sur leurs sorties sous un environnement de signaux stochastiques [83]. A un instant donné t, chaque signal en entrée, en interne, ou en sortie, est considéré comme une variable aléatoire prenant ses valeurs dans un ensemble de réalisations qui lui est propre (possiblement différent de celui des autres entrées). Chacune de ces variables aléatoires possède une densité de probabilité pouvant varier au cours du temps. Dans ce cas, les signaux sont qualifiés de processus aléatoires non-stationnaires.

Dans la réalité, un opérateur O possède un nombre fini K d'entrées représentées par un vecteur $(x_1, x_2, ..., x_K)_t$ à l'instant t. Ce vecteur est traité comme un vecteur dont les composantes sont des variables aléatoires, possiblement non stationnaires. Par opérateur, nous entendons par exemple un simple additionneur ou bien, à l'extrême, un système complexe tel qu'un système de communication. Les entrées peuvent être des entrées classiques en chemin de données comme les entrées d'un additionneur, ou bien des entrées constantes comme les coefficients constants d'un filtre LTI ou des paramètres de mise à jour adaptatifs. A l'instant t, une réalisation d'ensemble du vecteur aléatoire $(x_1, x_2, ..., x_K)_t$ devient un vecteur régulier appartenant à un domaine Ω_t qui consiste en l'ensemble global de réalisations possibles à l'instant t. Dans le cas où des signaux booléens Vrai et Faux sont traités comme des nombres réels tel que 0 et 1 respectivement, Ω_t est un sous ensemble de l'espace Euclidien R^K de dimension K. Le vecteur aléatoire $(x_1, x_2, ..., x_K)_t$ est considéré dans le domaine Ω_t :

$$(x_1, x_2, ..., x_K)_t \in \Omega_t. \tag{2.1}$$

Soit $t > t_1 > t_2 > ... > 0$ et supposons que le système commence à fonctionner à partir de l'instant t=0. A un instant t quelconque, la sortie d'un opérateur causal O dépend de toutes les entrées actuelles et précédentes $\{(x_1, x_2, ..., x_K)_t, (x_1, x_2, ..., x_K)_{t_1}, ..., (x_1, x_2, ..., x_K)_{t_N}, ..., (x_1, x_2, ..., x_K)_0\}$ et possiblement d'un état interne d'initialisation. Dans un système à échantillonnage régulier, l'instant t_i correspond simplement à $t - i$. Les variables internes décrivant l'état initial sont traitées comme des entrées additionnelles apportées au système par des additionneurs à l'instant 0. Par soucis de concision, nous ne tiendrons pas compte de cet état interne d'initialisation. La sortie de O est donnée par $f_O(x_1, x_2, ..., x_K, t)$ où f_O est la fonction de transfert de O pouvant s'exprimer par :

$$f_O(x_1, x_2, ..., x_K, t) = \Phi_{O,t}\left((x_1, x_2, ..., x_K)_t, (x_1, x_2, ..., x_K)_{t_1}, ..., (x_1, x_2, ..., x_K)_0\right). \tag{2.2}$$

La fonctionnalité de O à l'instant t est uniquement exprimée par la fonction $\Phi_{O,t}$ considérée comme la fonction de transfert instantanée de O à l'instant t. Dans ce cas le vecteur aléatoire des entrées de dimension $K\times(N+2)$ $\{(x_1, x_2, ..., x_K)_t, (x_1, x_2, ..., x_K)_{t_1}, ..., (x_1, x_2, ..., x_K)_{t_N}, (x_1, x_2, ..., x_K)_0\}$ est désigné par le vecteur $\{\xi_1, \xi_2, ..., \xi_M\}$ comme suit :

$$\begin{aligned} \xi_1 = x_1(t), \xi_2 = x_2(t), ..., \xi_K = x_K(t), \\ \xi_{K+1} = x_1(t_1), \xi_{K+2} = x_2(t_1), ..., \xi_{2K} = x_K(t_1), \\ ... \\ \xi_{M-K+1} = x_1(0), \xi_{M-K+2} = x_2(0), ..., \xi_M = x_K(0). \end{aligned} \tag{2.3}$$

avec $M = K \times (N + 2)$. Les quantités $(\xi_1, \xi_2, ..., \xi_M)$ sont appelées les variables étendues de $(x_1, x_2, ..., x_K)$ à l'instant t par rapport à l'opérateur O. Par conséquent la sortie de l'opérateur O à l'instant t est réduite à :

$$f_O(x_1, x_2, ..., x_K, t) = \Phi_{O,t}(\xi_1, \xi_2, ..., \xi_M). \tag{2.4}$$

En utilisant la relation (2.4), une valeur numérique de la sortie est disponible seulement avec une réalisation d'ensemble de vecteur aléatoire $(\xi_1, \xi_2, ..., \xi_M)$. Les variables étendues correspondent également à un domaine étendu :

$$(\xi_1, \xi_2, ..., \xi_M) \in \Omega_t \times \Omega_{t_1} \times ... \times \Omega_0. \tag{2.5}$$

Il est important de noter que f_O est une fonction déterministe. Lorsqu'un système est désigné physiquement, une fonction déterministe est souvent exécutée pour traiter certaines entrées aléatoires. Ceci est généralement accepté dans la modélisation des systèmes de traitement numérique de signal. Des exemples d'opérateurs (additionneur, multiplexeur, ...) sont manipulés selon cette modélisation dans [83].

En résumé, tous les facteurs aléatoires caractérisant la sortie d'un opérateur sont introduits par ses entrées aléatoires, alors que la fonction de transfert $\Phi_{O,t}$ est une fonction déterministe connue (comme une fonction du temps t).

2.1.2 Les opérateurs lisses et non-lisses

Le concept du caractère lisse d'un opérateur est à la base de la théorie des perturbations. Comme défini dans [83], une fonction déterministe $\Phi_{O,t}$ est considérée comme lisse sur les signaux quantifiés si elle est continue et dérivable de n'importe quel degré sur un ensemble ouvert dans lequel les signaux arithmétiques font partie, quelles que soient les réalisations des signaux logiques.L'opération O est appelée lisse sur ses entrées arithmétiques, ou brièvement lisse. Dans [83], il a été montré que la dérivabilité de degré 3 est toujours suffisante afin de modéliser les effets de quantifications. Nous pouvons remarquer que le caractère lisse ou non lisse d'un opérateur n'a pas de sens lorsque ses entrées correspondent à des signaux logiques. En revanche, des opérateurs de base tels que additionneurs ou multiplieurs sont lisses. Considérons, par exemple, l'opérateur I inverseur dont la fonction de transfert est donnée par :

$$f_I(x, t) = \Phi_{I,t}(\xi_1, ..., \xi_M) = 1/\xi_1. \tag{2.6}$$

Où ξ_1 est un signal arithmétique. Il est clair que $\Phi_{I,t}$ est lisse si ses entrées appartiennent à $]-\infty, 0[\cap]0, +\infty[$. Par conséquent, cette fonction est lisse sur ce domaine et elle est considérée comme non lisse dans $\mathbb{R}$ puisqu'elle n'est plus de classe $\mathcal{C}^1$. De même, l'opérateur valeur absolue est lisse sur le même domaine mais pas sur $\mathbb{R}$ puisqu'elle n'est pas de classe $\mathcal{C}^1$. En fait, la plupart des opérations mathématiques sont lisses sur leurs domaines de définition.

D'après ces définitions, les opérateurs de décision avec des entrées arithmétiques et des sorties logiques sont des opérateurs *non lisses*, mais, ils peuvent être traités comme lisses sur un domaine

de définition bien particulier où les entrées appartiennent à la même région de décision. Les régions non lisses contiennent les entrées arithmétiques qui produisent différentes décisions en ajoutant une perturbation infiniment petite. De plus, d'après [83], un opérateur est *non lisse* sur un domaine donné si sa fonction caractéristique n'est pas de classe $\mathcal{C}^1$ sur ce domaine.

2.1.3 La quantification d'un opérateur non lisse

En terme de quantification, nous pouvons considérer un opérateur comme *non lisse* si la déviation des statistiques actuelles de l'erreur issue de sa quantification est suffisamment importante devant les statistiques de l'erreur déterminée par le modèle PQN.

Il convient de noter que si la fonction caractéristique de la FDP du signal en entrée est à bande limitée, l'erreur d'estimation est toujours présente. L'amplitude de cette erreur augmente en fonction du pas de quantification. L'acceptabilité de l'erreur est donc un paramètre défini par l'utilisateur. Un compromis doit être trouvé entre (i) l'introduction d'une erreur de quantification importante ce qui permet de considérer certains opérateurs quantifiés dans le système comme étant lisses, et (ii) la maîtrise d'une erreur de quantification limitée qui rend la plupart des opérateurs comme non lisses. Par ailleurs, sachant que l'un des opérateurs en arithmétique virgule fixe est lisse, nous pouvons en déduire le caractère lisse d'autres opérateurs de la connaissance de la largeur de bande du signal dans le domaine de sa fonction caractéristique.

2.1.4 Identification d'un opérateur non lisse

Dans un scénario pratique, un système en arithmétique virgule fixe consiste en un certain nombre d'opérations de quantification. Les estimations analytiques obtenues en utilisant le modèle PQN peuvent être erronées lorsqu'elles sont appliquées sur la quantification d'un opérateur *non lisse*. Par conséquent, il est important de classifier les opérateurs comme lisses ou non lisses en se basant sur les caractéristiques du signal à quantifier et le pas de quantification. Nous proposons dans cette section une technique basée sur le calcul de la fonction caractéristique pour vérifier l'applicabilité du modèle PQN dans le cas de n'importe quel opérateur de quantification uniforme. Cette technique identifie les bornes du pas de quantification afin que le modèle PQM puisse être appliqué pour modéliser le comportement de l'erreur.

Cette stratégie considère tout d'abord la totalité des opérations de quantification impliquées dans le graphe du système G constitué par N_{op} opérateurs reliés par N_c connexions (figure 2.1).

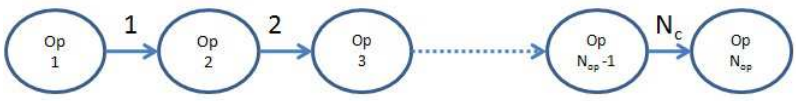

FIGURE 2.1 – Graphe du système

Un opérateur est représenté par l'un de ces nœuds et comporte des connexions multiples en entrée et une seule connexion en sortie. En cas d'implémentation utilisant une précision finie, une telle opération peut essentiellement être représentée par le schéma de l'opérateur suivi de l'opération de quantification. Les schémas de l'opération pour les deux précisions finie et infinie sont montrés par la figure 2.2.

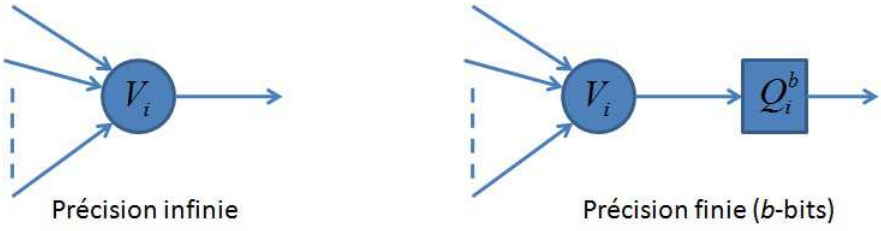

FIGURE 2.2 – Schéma d'un opérateur dans la précision infinie et la précision finie avec b bits

Dans le cas d'une implémentation en arithmétique virgule fixe, les entrées des opérateurs sont aussi quantifiées. L'effet de la quantification de l'entrée est couvert par les quantificateurs présents à la sortie du nœud opérateur à partir duquel le signal provient. Par conséquent, un quantificateur peut être associé avec chaque signal. L'effet des entrées quantifiées a un impact sur le comportement de l'erreur à la sortie de l'opérateur considéré. Dans un tel scénario, il est possible d'avoir les effets de la double quantification (annexe A). Cependant, une quantification répétée ne change pas fondamentalement la dynamique. Ainsi, pour évaluer le caractère lisse de chaque opérateur quantifié, il suffit de travailler avec les statistiques des données en double précision à tous les points d'intérêt. La technique développée est présentée dans cette section.

Cette technique commence par considérer que tous les opérateurs quantifiés avec un quantificateur Q_i sont lisses. Ensuite, des simulations en arithmétique virgule flottante, d'un ensemble exhaustif des vecteurs tests, sont utilisées afin de déterminer les caractéristiques de la FDP $f_{x_i}(x_i)$ du signal à l'entrée du i^{eme} quantificateur Q_i. D'autres techniques alternatives peuvent être utilisées pour estimer la FDP du signal comme par exemple les techniques analytiques présentées dans [75].

La seconde étape consiste en un test de déviation de l'erreur. Pour cela, il est tout d'abord nécessaire d'évaluer, soit analytiquement ou soit par simulation [92], le nombre minimal de bits b du quantificateur ($Q_i^b = b$) tels que la déviation ϵ_i^b du $i^{ème}$ signal d'entrée soit quasi nulle, soit donc que l'opérateur quantifié avec Q_i puisse être considéré comme lisse. Une fois ceci réalisé, l'utilisateur fixe un seuil τ_i correspondant à la tolérance autorisée sur l'erreur d'estimation. Ce seuil peut être spécifique à un quantificateur donné ou bien alors il peut être fixé de manière uniforme à une valeur donnée qui sera appliquée à l'ensemble des quantificateurs du système. Tant que la déviation par rapport au modèle PQN est plus petite que la tolérance à l'erreur ($|\epsilon_i^b| < \tau_i$), le nombre de bits affectés continue à être décrémenté par 1 et la déviation par rapport au modèle PQN ϵ_i^b est évaluée de nouveau pour la nouvelle longueur du mot de code affectée.

$$\epsilon_i^b = \textbf{Evaluer}(f_{x_i}, 2^{-b}). \tag{2.7}$$

Cette opération se répète sous forme d'une boucle. A chaque itération de la boucle, l'opérateur quantifié peut être considéré comme *non lisse*. Afin de réduire le nombre d'exécutions de la boule, il est également possible d'utiliser une variante de la méthode précédente qui consiste à remplacer l'opération de décrémentation binaire par une procédure de recherche binaire.

Lorsque la déviation de l'erreur est supérieure à la tolérance ($|\epsilon_i|^b \geq \tau_i$), cela signifie que l'opérateur quantifié qui était lisse jusqu'à l'étape de quantification précédente devient maintenant *non lisse*. En d'autres termes, cette transition marque la frontière entre le pas de quantification lisse et *non lisse* pour le signal associé au i^{eme} opérateur. Par conséquent, la valeur de $b+1$ est affectée à Q_b^i afin de marquer la plus petite longueur du mot de code en vertu de laquelle l'opérateur peut être considéré comme lisse.

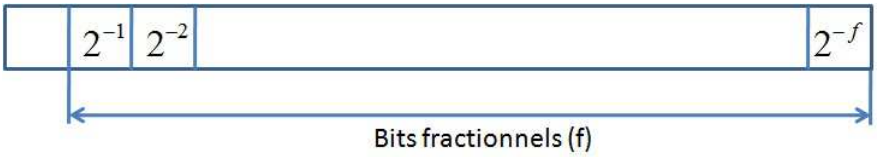

FIGURE 2.3 – Représentation d'un signal en virgule fixe avec une dynamique $[-1, 1)$

Afin de bien analyser le comportement de la déviation par rapport au modèle PQN, nous considérons un signal normalisé dans l'intervalle $[-1, 1]$. Son format virgule fixe est illustré par la figure 2.3. La figure 2.4 montre l'amplitude de l'erreur relative entre le modèle analytique PQN et les estimations de l'erreur obtenues par simulation. Nous considérons dans cette illustration trois FDP : la distribution uniforme, Laplacienne et Gaussienne. L'erreur relative entre le modèle PQN et la simulation est importante dans le cas Laplacien, et, dans une moindre mesure, pour une densité de probabilité gaussienne. Enfin, l'erreur relative devient faible, voire négligeable, dans le cas d'une loi uniforme. En effet, puisque la loi uniforme possède des valeurs réparties sur l'intervalle $[-1, 1]$, par conséquent, sa quantification (avec le plus grand nombre de bits) préserve des formes uniformes et les valeurs de -1 ou 1 contribuent à la puissance totale de bruit. Tandis que, dans le cas de la distribution gaussienne, les valeurs sont plus denses près de la valeur de 0 et donc remises à zéro lorsque des pas de quantification larges sont utilisés. Ceci engendre une déviation large entre l'erreur estimée par le modèle PQN et l'erreur observée par simulation. Ce phénomène est plus visible dans le cas de la distribution Laplacienne où il y a plus de valeurs proches de 0 que dans le cas Gaussien. Comme le pas de quantification est réduit (davantage de bits sont affectés pour la précision), les valeurs ne sont pas remises à 0 et donc convergent à un comportement similaire.

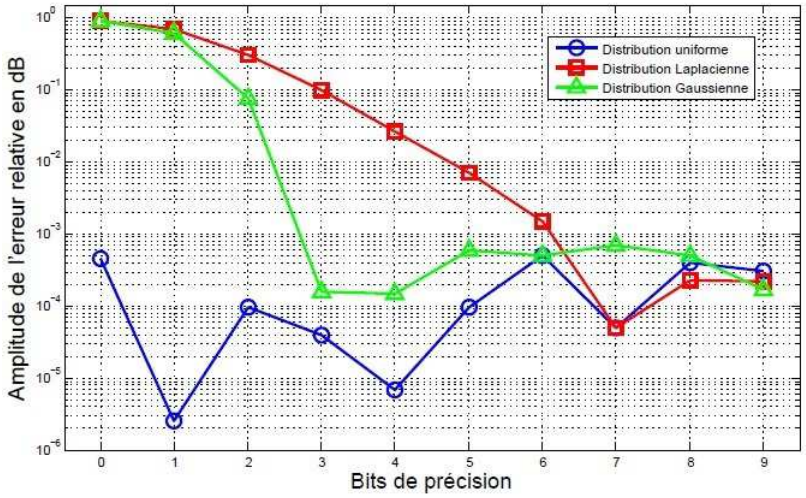

FIGURE 2.4 – Représentation d'un signal en virgule fixe avec une dynamique $[-1, 1)$

Dans l'algorithme décrit précédemment, il est nécessaire d'enregistrer les statistiques de la valeur prise par chaque signal associé à un opérateur de quantification dans le système. En effet, la mémorisation des valeurs de tous les signaux à travers la totalité de la longueur de simulation peut consommer une grande quantité de mémoire et de temps, ce qui peut parfois conduire à une infaisabilité en pratique. Pour remédier à ce problème, il est possible de ne mémoriser qu'uniquement les statistiques des signaux associés à des opérateurs quantifiés soupçonnés d'être non-lisses.

2.2 Modèle analytique pour l'opérateur de décision

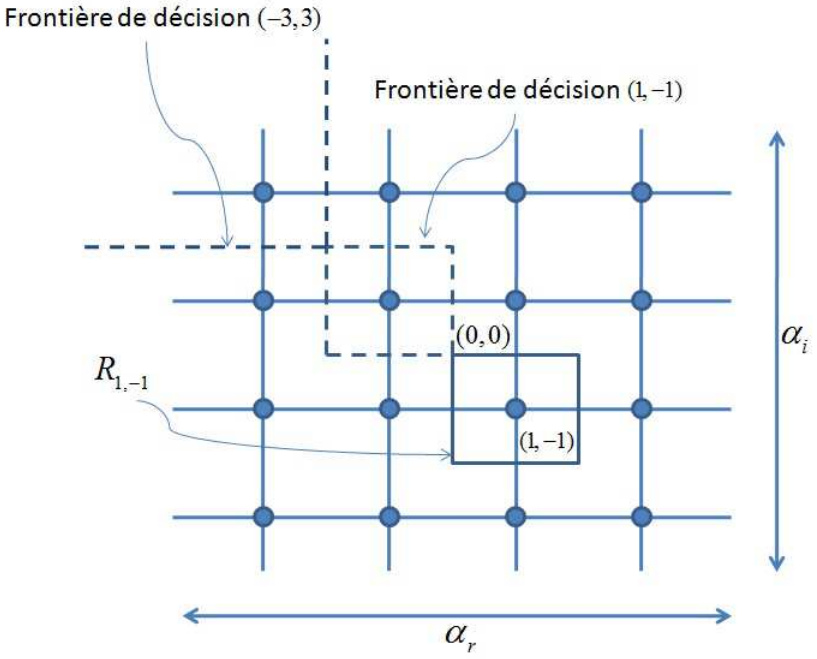

FIGURE 2.5 – Diagramme de la constellation 16-QAM

Il est fréquent de trouver un certain nombre d'opérateurs *lisses* et *non lisses* qui coexistent au sein des systèmes de traitement numérique de signal. Dans les systèmes de communication numérique, les opérateurs correspondent le plus souvent à des opérateurs de décisions QAM présents au sein des algorithmes de réception et ils coexistent avec d'autres opérateurs lisses en arithmétique virgule fixe. Citons à titre d'exemple les algorithmes de décodage sphérique tels le SSFE qui fera l'objet de la dernière partie de ce chapitre et et qui peut être considéré comme une application typique de la propagation du bruit de quantification lors d'une cascade d'opérateurs de décision. Il y a également les algorithmes d'égalisation tels que l'égaliseur à retour de décision qui sera présenté dans le prochain chapitre de cette thèse et qui est constitué d'une itération d'opérateurs de décision. Nous pouvons citer également les algorithmes de turbo décodage abordés dans le dernier chapitre.

Bien que les erreurs lisses puissent être estimées en utilisant le modèle additif PQN, la réponse d'un opérateur de décision QAM à une perturbation par le bruit de quantification peut être non linéaire ce qui peut provoquer un large écart par rapport aux estimations obtenues par l'application du modèle de PQN. Par conséquent, il est nécessaire d'analyser l'erreur causée par une quantification d'un opérateur *non lisse* et estimer son effet sur la performance d'un système. En particulier, étant donné que les opérateurs *lisses* coexistent avec les opérateurs *non lisses*, il est nécessaire d'estimer les statistiques de l'erreur à la sortie d'un opérateur *non lisse* en raison de la perturbation du signal à son entrée. Dans cette partie, nous commençons par modéliser l'opérateur de décision ainsi que sa réponse à une perturbation et proposer un modèle analytique permettant de déterminer la probabilité d'erreur de décision causée par la quantification d'un signal en virgule fixe. Il convient de préciser également que ce travail a été fait en collaboration avec les travaux de recherche menés par K. Parashar [75] dans la même équipe CAIRN.

2.2.1 Modélisation de l'opérateur de décision

Considérons le cas général d'un opérateur de décision au sein d'un système de communication numérique. Il s'agit d'un exemple typique d'un opérateur *non lisse*. La figure 2.5 montre la

constellation 16-QAM avec les différentes régions correspondant à chaque symbole de la constellation.

Le principe de fonctionnement d'un opérateur de décision QAM se base sur la différenciation des valeurs d'un signal QAM donné x à l'entrée de cet opérateur. Selon la valeur du signal à l'entrée, cet opérateur affecte à la sortie un symbole S_i représentant la région R_i dans laquelle se situe la valeur du signal à l'entrée. Chaque région R_i possède une limite r_i. Le symbole S_i appartient à l'ensemble des symboles de la constellation $S = \{S_1, S_2, ..., S_L\}$, avec $S_i \in \mathcal{R}^N$. N est la dimension du signal transmis ($N = 2$ pour les signaux QAM). Le fonctionnement de l'opérateur de décision est défini par la figure 2.6. Par conséquent la sortie $\tilde{x}$ de l'opérateur de décision est définie comme suit :

$$\tilde{x} = S_i \quad \textbf{si} \quad x \in R_i \tag{2.8}$$

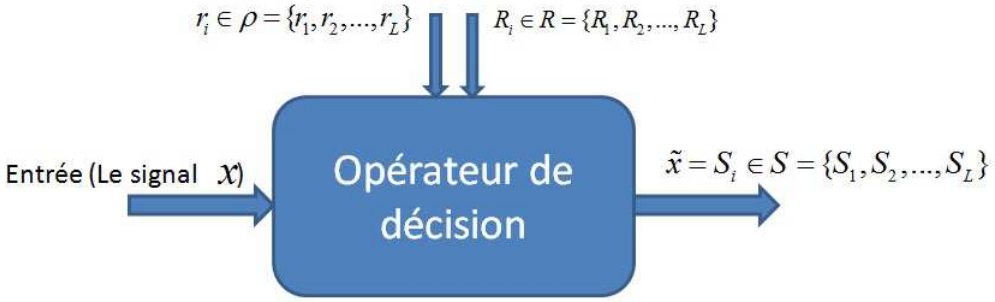

FIGURE 2.6 – Fonctionnement de l'opérateur de décision

R_i est la région définie comme le voisinage du point de la constellation S_i. La région R_i est un sous-espace de $\mathbb{R}^N$ et elle consiste essentiellement à l'ensemble des valeurs prises par x qui peuvent être décidées par le symbole S_i en sortie. Dans la figure 2.5, la région $R_{1,-1}$ est l'aire couverte par le rectangle centré sur le point de constellation $(1, -1)$.

$$R_i = \{x \in \mathbb{R}^2 / i = \underset{k}{\operatorname{argmin}} |x - S_k|^2\} \tag{2.9}$$

Les frontières des symboles de la constellation QAM $(-1, 1)$ et $(-3, 3)$ sont définies par les lignes discontinues. Les aires formées par ces frontières forment les régions $R_{-1,1}$ et $R_{-3,3}$.

En outre, l'opérateur de décision QAM porte beaucoup de similitudes avec la quantification rencontrée lors de l'étude de l'arithmétique de précision finie. Son fonctionnement est identique sur la voie en phase (axe réel ou horizontal) et sur la voie en quadrature (axe imaginaire ou vertical). Le comportement de l'opérateur QAM, restreint à un seul de ces axes (phase ou quadrature), peut être considéré comme un quantificateur Q de pas de quantification q en cascade avec un opérateur de saturation S de dynamique a comme indiqué sur la figure 2.7.

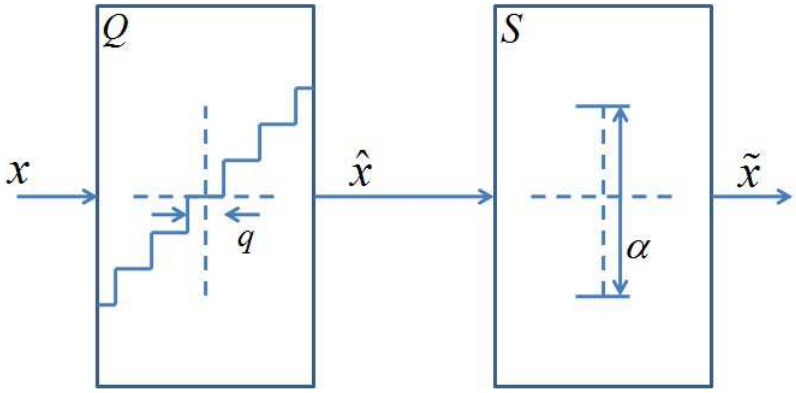

FIGURE 2.7 – Modèle de quantification d'un opérateur de décision

Le quantificateur Q suit le mode par arrondi et il est centré sur les points de la constellation. La dynamique α correspond à la différence maximale entre deux points de la constellation. Lorsque la dynamique du signal en entrée est comparable à la dynamique α de l'opérateur de saturation S (c-à-d $max(x) - min(x) \simeq \alpha$), l'opérateur de saturation peut être ignoré (c-à-d $\hat{x} \simeq \tilde{x}$) et l'opérateur de décision peut être approximé à un quantificateur.

Lorsque le comportement de l'opérateur de décision QAM est similaire au cas d'un quantificateur qui est utilisé pour modéliser l'arithmétique en précision finie, les statistiques de l'erreur sont *non lisses*. La différence entre l'opérateur de décision QAM et le quantificateur lisse qui est habituellement rencontré dans l'implémentation en précision finie consiste en un large pas de quantification par rapport au signal. Lorsque le pas de quantification lié à la précision finie est plus faible que la dynamique, le pas de quantification de l'opérateur de décision est souvent comparable au signal lui-même. Par conséquent, l'opérateur de décision est *non lisse*.

2.2.2 La réponse à la perturbation

Dans le cas d'un opérateur *non lisse*, les statistiques de l'erreur en sortie ne sont plus compatibles avec le modèle PQN. Par conséquent, l'impact du bruit de quantification à la sortie d'un opérateur *non lisse* peut être analysé en comparant uniquement la réponse de l'opérateur *non lisse* aux valeurs en entrée dans le cas d'une précision finie et leurs équivalents dans le cas de la précision infinie. L'effet de la perturbation causée par l'accumulation du bruit de quantification à l'entrée d'un opérateur *non lisse* est illustré par la figure 2.8.

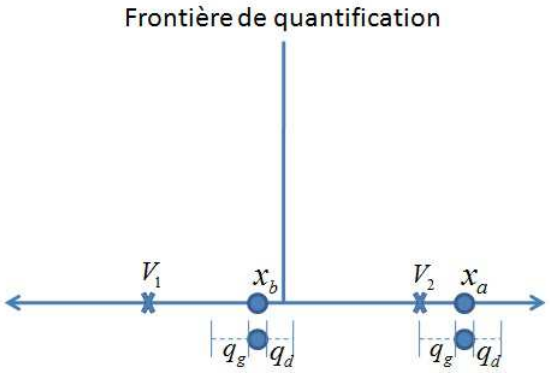

FIGURE 2.8 – Réponse à la perturbation à la frontière *non lisse* ; Cas A : $x = x_a$, Cas B :$x = x_b$

Soit un signal x à l'entrée d'un opérateur *non lisse* dont les sorties sont les valeurs V_1 ou V_2 dépendent de la valeur de x. Considérons deux scénarios dans lesquels un signal x prend

des valeurs x_a et x_b en précision infinie. En virgule fixe, le signal x est perturbé par le bruit de quantification accumulé. Soient q_g et q_d les distorsions négatives et positives probables maximales. Dans le cas où le signal prend une valeur suffisamment loin de la frontière tel que décrit dans le *cas A* : (soit $x = x_a$), l'amplitude de la perturbation q_g ou q_d n'est pas assez grande pour que le signal résultant puisse franchir la frontière de décision. Par conséquent, la valeur attribuée par l'opérateur non lisse, à la fois en précision finie et infinie, est V_2 et aucune erreur ne se propage. D'autre part, si une valeur en double précision x est assez proche de la frontière tel que décrit dans le *cas B* : (soit $x = x_b$), une perturbation positive peut conduire la valeur en virgule fixe à franchir la frontière de décision, soit donc à changer la valeur de la sortie en V_2 (à la place de V_1) provoquant ainsi un comportement de l'implémentation en précision finie différent de celui en précision infinie.

2.2.3 La probabilité d'erreur de décision

Pour analyser l'impact de la quantification, nous utilisons une métrique de précision qui compare les valeurs de divers signaux dans le système lorsque des opérateurs en virgule fixe sont utilisés. En cas de quantificateurs lisses, les erreurs sont faibles et donc facilement captées par le bruit de quantification. Dans le cas des opérateurs *non lisses*, la puissance des erreurs n'est pas conforme avec le modèle d'erreur analytique PQN. Par conséquent, il devient important d'avoir accès à la fonction de densité de probabilité de l'erreur en sortie de l'opérateur non lisse, ce qui permet également de déduire la puissance de l'erreur. En outre, il est plus fondamental de calculer la probabilité de l'erreur à la sortie de l'opérateur que de déduire directement la puissance du bruit du signal d'erreur de quantification.

Probabilité d'erreur de décision

Afin d'étudier l'impact du bruit de quantification *non lisse* accumulé à la sortie d'une première décision, nous considérons un système de traitement de signal (figure 2.9) formé d'un opérateur de décision à la sortie $\hat{x}$ d'un système lisse. Cette configuration est très fréquente dans les récepteurs des systèmes de communication numérique et dans les algorithmes de décodage et d'égalisation.

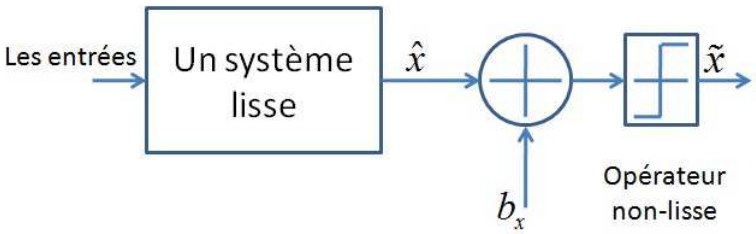

FIGURE 2.9 – Un système constitué d'un opérateur *non lisse*

La valeur d'un signal en virgule fixe $\hat{x}_{vfix}(n)$ à l'entrée de l'opérateur de décision est obtenue par la perturbation de $\hat{x}_{vflo}(n)$ à cause du bruit de quantification $b_x(n)$. Par conséquent le signal en entrée à l'opérateur *non lisse* de décision s'écrit sous la forme suivante :

$$\hat{x}_{vfix}(n) = \hat{x}_{vflo}(n) + b_x(n) \tag{2.10}$$

La probabilité d'erreur de décision causée par le bruit additif de quantification peut être déduite en comparant la sortie en virgule fixe $\tilde{x}_{vfix}(n)$ de l'opérateur de décision à la sortie en virgule flottante $\tilde{x}_{vflo}(n)$. Comme mentionné dans la section précédente, la déviation dans la valeur du signal dans un système en virgule fixe est fonction de la puissance du bruit de quantification associé au signal. Lorsque la valeur du signal est proche de la limite (frontière) de décision et que la puissance du bruit de quantification est suffisamment large, la déviation causée par la perturbation provoque le franchissement de la frontière de décision ce qui génère une sortie

différente de celle produite par un système avec une précision en virgule flottante. Par ailleurs, si la puissance du bruit est suffisamment grande, la déviation de la valeur du signal peut traverser plusieurs frontières ce qui engendre inévitablement une erreur de décision à la sortie.

Soit $P_{i,j}$ la probabilité d'erreur de décision $(i \neq j)$ telle que $\tilde{x}_{vflo} = S_i$ et $\tilde{x}_{vfix} = S_j$.

Cette erreur correspond au cas où la valeur entrée (située dans la région R_i, est perturbée par le bruit b_x de telle sorte que le signal résultant appartienne à la région R_j.

Par conséquent, la probabilité d'erreur de décision $P_{i,j}$ est la probabilité que le signal en entrée en virgule flottante $\hat{x}_{vflo}$ appartienne à la région R_i et que le signal correspondant en virgule fixe $\hat{x}_{vfix}$ appartienne à la région R_j.

$$P_{i,j} = \mathbf{Prob}\left\{(\hat{x}_{vflo} \in R_i) \cap (\hat{x}_{vflo} + b_x \in R_j)\right\}. \tag{2.11}$$

Soient $f_b(b)$ et $f_x(x)$ les fonctions densité de probabilité respectivement du bruit b_x et du signal d'entrée en arithmétique virgule flottante $\hat{x}_{vflo}$. La probabilité P_i que la valeur du signal en entrée $\hat{x}_{vflo}$ donne une décision $\tilde{x}_{vflo} = S_i$ est obtenue en intégrant la FDP $f_x(x)$ sur toute la région du symbole correspondant R_i et elle est donnée par :

$$P_i = \int_{R_i} f_X(x)dx \tag{2.12}$$

Considérons une instance du signal x qui prend la valeur $x_i \in R_i$ telle que la sortie est $\tilde{x} = S_i$. En connaissant la FDP du bruit de quantification b_x, il est possible de calculer la probabilité d'erreur $P_{x_i,j}$ causée par la perturbation du signal avec une valeur x_i qui conduit à une décision du symbole Sj (j différent de i) en précision finie en raison du bruit de quantification bx, alors que cette décision correspond au symbole Si en arithmétique virgule flottante. Par conséquent, $P_{x_i,j}$ est définie par :

$$P_{x_i,j} = \mathbf{Prob}\left\{\hat{x}_{vflo} + b_x \in R_j | \hat{x}_{vflo} \in R_i\right\} \tag{2.13}$$

En supposant que le bruit de quantification est non corrélé avec le signal [92], la relation précédente peut encore s'exprimer selon :

$$P_{x_i,j} = \lim_{\epsilon \to 0} \int_{x_i - \frac{\epsilon}{2}}^{x_i + \frac{\epsilon}{2}} f_x(x_i)dx_i . \int_{R_j} f_b(b - x_i)db. \tag{2.14}$$

La probabilité d'erreur de décision totale $P_{i,j}$ est obtenue en considérant tous les points de la région R_i et en intégrant la probabilité $P_{x_i,j}$ sur la région R_i :

$$P_{i,j} = \int_{R_i} P_{x_i,j}dx_i \tag{2.15}$$

En remplaçant $P_{x_i,j}$ par son expression de l'équation (2.14), la probabilité d'erreur de décision s'écrit :

$$P_{i,j} = \int_{R_i} \left[f_x(x_i) \int_{R_j} f_b(b - x_i)db \right] dx_i. \tag{2.16}$$

En arrangeant les termes de l'équation précédente, la probabilité d'erreur de décision peut s'écrire sous la forme suivante :

$$P_{i,j} = \int_{R_i} \int_{R_j} f_x(x_i) f_b(b - x_i)dbdx_i. \tag{2.17}$$

A partir de cette équation, l'influence du bruit de quantification et du signal à l'entrée de l'opérateur de décision peuvent être analysées sur chaque région de décision. Pour ce faire, considérons la fonction de densité de probabilité $f_{T_i}(b)$ du signal présent à l'entrée de l'opérateur de décision et tel que x_i appartient à R_i lorsque l'on utilise une arithmétique en virgule flottante.

Pour chaque région R_i, la FDP de la contribution totale du bruit de quantification est donnée par :

$$f_{T_i}(b) = \int_{R_j} f_x(x_i) f_b(b - x_i)dx_i. \tag{2.18}$$

La probabilité d'erreur de décision $P_{i,j}$ peut s'écrire en fonction de $f_{T_i}(b)$ comme suit :

$$P_{i,j} = \int_{R_j} f_{T_i}(b)db. \tag{2.19}$$

En calculant toutes les probabilités $P_{i,j}$, nous obtenons la matrice des probabilités P dont les coefficients sont les $P_{i,j}$:

$$(P_{i,j}) = P_{\tilde{x}_{vflo},\tilde{x}_{vfix}}(\tilde{x}_{vflo} = S_i, \tilde{x}_{vfix} = S_j). \tag{2.20}$$

La distribution de probabilité du signal réel dans le cas d'un bruit en virgule fixe peut être obtenue sous la forme d'une distribution marginale de la distribution conjointe obtenue jusqu'à présent.

Fonction de probabilité jointe

Étant donné que la sortie de l'opérateur de décision est discrète, la fonction de densité de l'erreur s'exprime directement en fonction des valeurs de l'ensemble S des symboles de décision. Nous pouvons alors introduire la distance $d_{i,j}$ séparant deux symboles de l'alphabet S :

$$d_{i,j} = S_i - S_j. \tag{2.21}$$

Les erreurs individuelles de décision sont représentées par une fonction discrète qui s'exprime selon qui la fonction $\delta(\tilde{x})$ qui la fonction delta de Kronecker. Par conséquent la fonction de probabilité jointe est obtenue par :

$$f_{\tilde{x}}(\tilde{x}) = \sum_{i=1}^{L}\sum_{j=1}^{L} P_{i,j}.\delta(\tilde{x} - d_{i,j}) \tag{2.22}$$

La fonction $\delta(\tilde{x} - d_{i,j})$ est une version décalée de la fonction δ dans l'espace bidimensionnel. Cette densité de probabilité est caractérisée par $L \times L$ valeurs discrètes correspondant aux différentes permutations de l'ensemble des symboles décision S. Chaque $d_{i,j}$ $(i \neq j)$ correspond à une erreur de décision générée pour chaque couple (i, j) et la probabilité d'erreur de décision correspondante est $P_{i,j}$ obtenue par l'équation (2.17).

Afin de valider ce modèle de fonction de probabilité, nous pouvons utiliser deux vérifications. La première analyse consiste à vérifier théoriquement que la somme de toutes les probabilités individuelles $P_{i,j}$ est égale à 1. Comme le signal x est indépendant du bruit de quantification b, la somme peut s'écrire sous cette forme.

$$\sum_{i=1}^{L}\sum_{j=1}^{L} P_{i,j} = \sum_{i=1}^{L}\int_{R_i} f_x(x_i).\sum_{j=1}^{L}\int_{R_j} f_b(b - x_i).db.dx_i \tag{2.23}$$

Par définition, l'aire sous une FDP est toujours égale à 1. Par conséquent, la somme interne $(\sum_{j=1}^{L}\int_{R_j} f_B(b - x_i).db.dx_i)$ est égale à l'unité car elle correspond à l'intégration de la FDP du bruit de quantification sur l'ensemble $\cup R_j$.

En suivant le même raisonnement, la somme extérieure $(\sum_{i=1}^{L}\int_{R_i} f_X(x_i))$ est également égale à l'unité. En conséquence, la double somme (2.23) est elle aussi égale à l'unité tout comme la fonction de densité de probabilité (2.22). Par conséquent, nous pouvons conclure que la fonction de probabilité donnée par 2.22 est une fonction densité de probabilité valide.

D'autre part, nous pouvons également déterminer la probabilité d'erreur de décision totale P_{erreur} qui est la somme de toutes les probabilités d'erreur de décision individuelle $P_{i,j|i\neq j}$.

$$P_{erreur} = \sum_{i=1}^{L}\sum_{j\neq i,j=1}^{L} P_{i,j} \quad \forall(i,j) \in [1, L] \tag{2.24}$$

Cette probabilité d'erreur totale peut être déterminée à partir des probabilités individuelles $P_{i,i}$ qui sont les probabilités de ne pas avoir une erreur de décision. En d'autres termes, la probabilité $P_{i,i}$ est la probabilité que la sortie de l'opérateur de décision dans le cas de virgule fixe et dans le cas de virgule flottante soit égale au symbole S_i. Par conséquent, la probabilité d'erreur de décision totale s'exprime comme suit :

$$P_{erreur} = 1 - \sum_{i=1}^{L} P_{i,i} \tag{2.25}$$

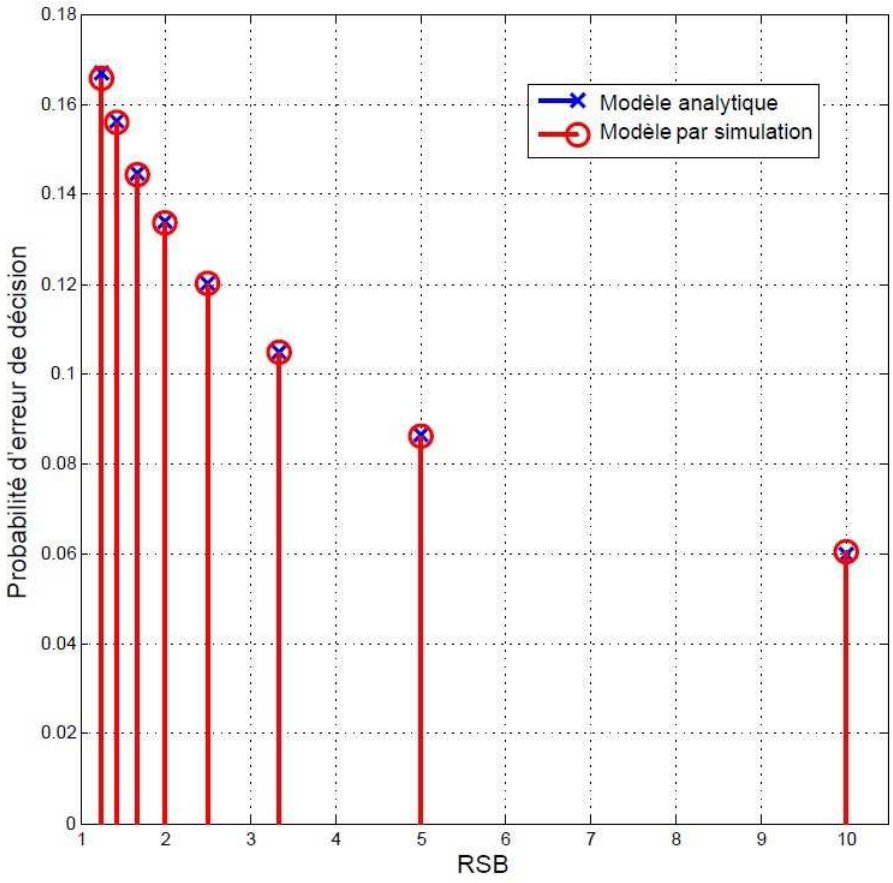

FIGURE 2.10 – Probabilité d'erreur de décision totale : cas BPSK

La deuxième vérification se base sur la comparaison des probabilités $P_{i,j}$ obtenues analytiquement par ce modèle avec celles obtenues par simulations. Par conséquent, nous considérons un système tel que décrit dans la figure 2.9 qui est constitué d'un bloc lisse suivi d'un opérateur de décision *non lisse*. Nous supposons que le signal présent à l'entrée de l'opérateur de décision est un signal gaussien BSPK centré sur les symboles avec une équiprobabilité entre les différents symboles de la constellation (-1, 1) et de variance σ_{ch}^2. Dans ce cas nous avons deux probabilités d'erreur de décision causée par le bruit de quantification b. Ces probabilités sont $P_{-1,1}$ qui est la probabilité d'avoir en sortie le symbole -1 en virgule flottante alors qu'en virgule fixe nous obtenons le symbole 1 et $P_{1,-1}$ est la probabilité du cas contraire. Afin de tester la fiabilité de ce modèle, nous supposons que le bruit en entrée de l'opérateur suit une loi gaussienne de moyenne nulle et de variance σ_b^2. Les FDP du signal à l'entrée de l'opérateur de décision et du bruit de quantification sont données par :

$$f_X(x) = \frac{1}{2\sqrt{2\pi}\sigma_{ch}} \left(\exp(-\frac{(x-1)^2}{2\sigma_{ch}^2}) + \exp(-\frac{(x+1)^2}{2\sigma_{ch}^2}) \right). \tag{2.26}$$

$$f_B(b-x) = \frac{1}{\sqrt{2\pi}\sigma_b} \exp(-\frac{(b-x)^2}{2\sigma_b^2}). \tag{2.27}$$

En appliquant le modèle analytique de l'équation (2.17), nous obtenons les probabilités d'erreur de décision $P_{-1,1}$ et $P_{1,-1}$. En raison de l'hypothèse d'équiprobabilité entre les deux symboles de la constellation, les deux probabilités d'erreur de décision $P_{-1,1}$ et $P_{1,-1}$ sont égales. Par conséquent, la probabilité d'erreur totale P_{erreur} est le double de $P_{1,-1}$. Afin de tester le modèle, nous varions la variance σ_b^2 de 0.1 jusqu'à 0.8 et nous calculons à chaque fois la probabilité d'erreur de décision totale analytiquement par le modèle obtenu et par simulation. Le calcul par simulation se base sur une simulation en virgule fixe et une simulation en virgule flottante et la comparaison des sorties de l'opérateur de décision dans les deux cas. Si ces deux sorties sont différentes, il s'agit d'une erreur due à la conversion en virgule fixe où en d'autres termes à la perturbation causée par le bruit de quantification additif b. La figure 2.10 présente la probabilité d'erreur de décision obtenue analytiquement et par simulation en fonction de la variance du bruit de quantification. Il convient de préciser que, dans cette expérience, nous modélisons la conversion en virgule fixe par l'ajout du bruit de quantification au signal à l'entrée de l'opérateur. Par contre dans les prochaines expériences nous effectuerons une "vraie" conversion en format virgule fixe et par conséquent chaque RSB correspondra à un format virgule fixe bien déterminé. Nous remarquons d'après la figure 2.10, que les résultats obtenus par le modèle analytique de l'équation (2.17) correspondent fidèlement aux résultats obtenus par simulations. L'erreur maximale entre la probabilité obtenue par simulation et celle obtenue analytiquement est de l'ordre de 1%.

2.3 Cascade d'opérateurs de décision

Dans le paragraphe précédent, nous avons proposé un modèle analytique permettant d'estimer la probabilité d'erreur de décision d'un opérateur de décision dans le cas particulier d'un système composé d'un bloc lisse et d'un unique opérateur de décision. Cependant, ce modèle peut-il être appliqué à un système composé de plusieurs opérateurs de décision ? Si l'on considère un tel système, l'erreur causée par la quantification au niveau du premier opérateur de décision se propage tout au long du système en passant par différents blocs pour arriver au deuxième opérateur de décision. Le modèle analytique d'estimation de la probabilité d'erreur de décision n'est plus applicable pour estimer l'erreur de décision au niveau du deuxième opérateur de décision car ce modèle ne tient pas compte de la propagation de l'erreur due à la quantification et au format virgule fixe.

Afin de bien illustrer cette problématique, nous considérons un système composé de deux opérateurs de décision tels que montré par la figure 2.11.

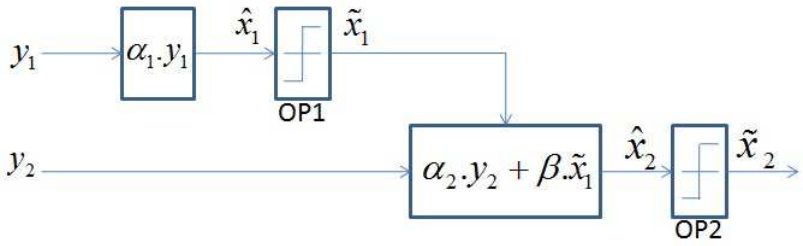

FIGURE 2.11 – Cascade de deux opérateurs de décision

Le système considéré est un cas simplifié d'un système de décodage sphérique composé de deux antennes. Ce système sera étudié d'une façon approfondie dans la prochaine section. D'après le diagramme bloc du système, nous remarquons que l'entrée $\hat{x}_2$ du deuxième opérateur de décision OP_2 ne dépend pas seulement du signal reçu y_2 mais aussi de la sortie $\tilde{x}_1$ du premier opérateur de décision. Lorsqu'une erreur de quantification est produite au niveau du premier opérateur,

elle se propage jusqu'au deuxième opérateur de décision. Si nous essayons d'appliquer le modèle analytique de l'équation (2.17), nous obtenons des estimations de l'erreur de décision très éloignées des estimations obtenues par simulation. La figure 2.12 représente la probabilité d'erreur de décision estimée au niveau du deuxième opérateur de décision par le modèle analytique à un seul opérateur de décision et également par simulation.

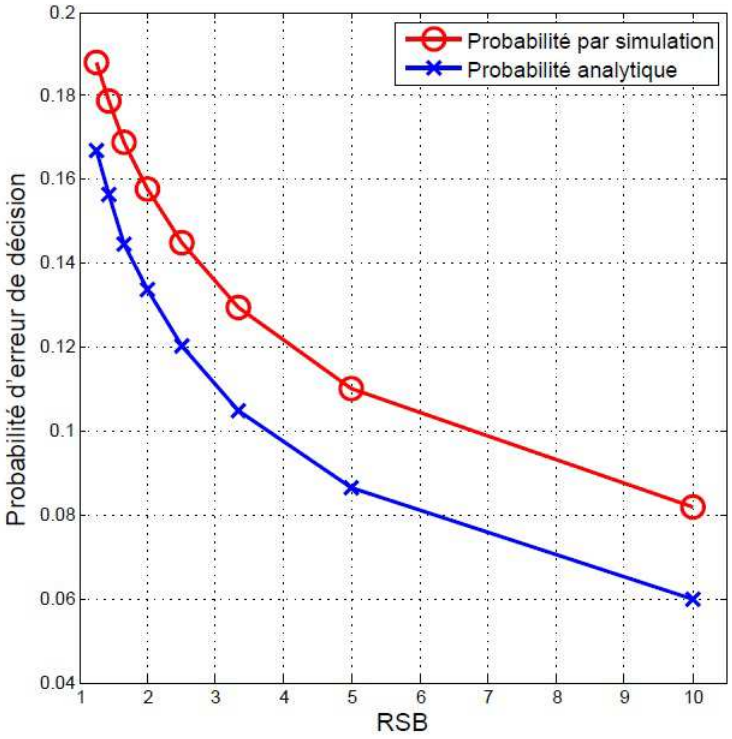

FIGURE 2.12 – Non validité du modèle analytique d'estimation de la probabilité d'erreur de décision dans le cas d'une cascade d'opérateurs de décision

Nous pouvons remarquer que la probabilité d'erreur de décision obtenue par le modèle analytique est sous-estimée par rapport à celle obtenue à partir de la simulation. Comme le modèle analytique proposé ne tient pas compte de la propagation des erreurs de quantification, les résultats de la figure 2.12 étaient prévisibles. Par conséquent, il est fondamental d'étudier la propagation de l'erreur de quantification dans un système composé de plusieurs opérateurs *non lisses* et de tenir compte de cette propagation. Ceci sera présenté dans la prochaine section.

2.3.1 Propagation de l'erreur de quantification

Pour valider ce modèle dans le cas d'une cascade d'opérateurs de décision, il est absolument nécessaire de prendre en compte le phénomène de propagation des erreurs de quantification. Considérons le système présenté à la figure 2.13 composé à la fois de plusieurs opérateurs lisses et de $N+1$ opérateurs *non lisses*.

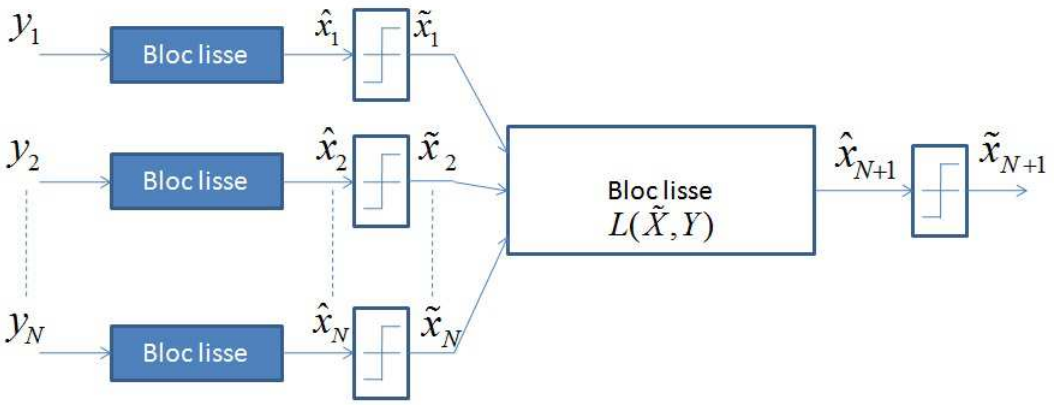

FIGURE 2.13 – Propagation de l'erreur de quantification dans un système composé de plusieurs opérateurs *non lisses* en cascade

Dans le cas d'une implémentation en virgule fixe, le vecteur des entrées $Y = [y_1, y_2, ..., y_N]$ est perturbé par le vecteur des bruits de quantification $B = [b_1, b_2, ..., b_N]$, chaque bruit b_i perturbant le signal y_i. présent sur la ième entrée. Soient Y_{vflo} et Y_{vfix} les vecteurs des entrées respectivement en virgule flottante et en virgule fixe. L'opérateur L représente une fonction lisse comportant $2N$ entrées correspondant au vecteur Y et au vecteur $\tilde{X} = [\tilde{x}_1, \tilde{x}_1, ..., \tilde{x}_N]$ constitué des N sorties des opérateurs de décision situés en amont. De même, soient $\tilde{X}_{vflo}$ et $\tilde{X}_{vfix}$ les vecteurs des sorties des N opérateurs de décision qui précèdent le bloc lisse respectivement en arithmétique virgule flottante et en virgule fixe. Soient $\hat{x}_{N+1}^{vflo}$ et $\hat{x}_{N+1}^{vfix}$ l'entrée au dernier opérateur de décision respectivement en représentation virgule flottante et virgule fixe. Par conséquent, la sortie du bloc lisse en virgule fixe $\hat{x}_{N+1}^{vfix}$ est donnée par :

$$\begin{aligned} \hat{x}_{N+1}^{vfix} &= L(y_1^{vfix}, ..., y_N^{vfix}, \tilde{x}_1^{vfix}, ..., \tilde{x}_N^{vfix}) \\ &= L(y_1^{vflo}, ..., y_N^{vflo}, \tilde{x}_1^{vflo}, ..., \tilde{x}_N^{vflo}) \\ &+ l(b_1, ..., b_N, \tilde{x}_1^{vflo}, ..., \tilde{x}_N^{vflo}, \tilde{x}_1^{vfix}, ..., \tilde{x}_N^{vfix}) + b_L \\ &= \tilde{x}_{N+1}^{vflo} + L_b(B, \tilde{X}_{vflo}, \tilde{X}_{vfix}) + b_L \end{aligned} \tag{2.28}$$

b_L est le bruit ajouté par les quantificateurs lisses lors de l'utilisation d'une implémentation en virgule fixe de la fonction L et L_b est la fonction de la propagation du bruit.

Étant donné que la fonction L est lisse, la fonction de propagation du bruit L_b est linéaire par rapport au vecteur des bruits de perturbation b. La propagation des erreurs à la sortie des N opérateurs de décision *non lisses* à travers la fonction L peut ne pas avoir des caractéristiques linéaires.

Par conséquent, la fonction de propagation du bruit L_b doit être évaluée pour chaque erreur et pour toutes les configurations d'erreurs présentes sur les sorties des N opérateurs de décision situés en amont.

2.3.2 Détermination analytique de la probabilité d'erreur en sortie de la cascade

Soit v_{vflo} et v_{vfix} deux instances des valeurs prises par $\tilde{x}_{vflo}$ et $\tilde{x}_{vfix}$ respectivement. Le signal $\hat{x}_{N+1}^{vfix}$ est donné par :

$$\begin{aligned} \hat{x}_{N+1}^{vfix} &= \hat{x}_{N+1}^{vflo} + L_b^v(B) + b_L \\ &= \hat{x}_{N+1}^{vflo} + b_f \end{aligned} \tag{2.29}$$

La fonction de propagation du bruit L_b^v est déduite de la fonction L_b de l'équation (2.28) en utilisant les valeurs particulières v_{vflo} et v_{vfix} à la place de $\tilde{X}_{vflo}$ et $\tilde{X}_{vfix}$, respectivement. Par conséquent, la fonction L_b ne varie qu'en fonction du seul paramètre B correspondant au vecteur des perturbations. Ainsi, la propagation des perturbations associées aux signaux en entrées Y à travers la fonction L est caractérisée par une combinaison spécifique donnée des vecteurs de données en entrée de décision en format virgule flottante et virgule fixe. De manière équivalente, nous pouvons considérer que les vecteurs de données en entrée de décisions en arithmétique virgule fixe et virgule flottante représentent conjointement un scénario S des conditions en entrée. En sortie de la fonction L, le bruit généré par les opérations en virgule fixe dans l'implémentation de la fonction L et la propagation des perturbations en entrée peut être résumé et collecté à travers le bruit b_f. Soit M_k le nombre des symboles de décision disponibles à la sortie du k^{eme} opérateur de décision. Le vecteur $\tilde{X}$ peut prendre $C = \prod_{k=1}^{N} M_k$ combinaisons possibles. Si l'on considère le vecteur des valeurs particulières S_N^k défini par $S_N^k = [v_{vflo}, v_{vfix}]$, il existe $C \times C$ combinaisons possibles pour ce vecteur

La probabilité d'erreur de décision $P_{i,j}^{S_N^k}$ à la sortie du $(N+1)^{eme}$ opérateur de décision pour une combinaison donnée du vecteur des entrées S_N^k peut être calculée en utilisant le modèle analytique de l'équation (2.17). $P_{i,j}^{S_N^k}$ définit la probabilité que la valeur en sortie du $(N+1)^{ème}$ opérateur de décision $\tilde{x}_{N+1}$ corresponde à S_i en format virgule flottante et à S_j en arithmétique virgule fixe sous la contrainte du scénario défini par les entrées de décision S_N^k, elle est donnée par :

$$P_{i,j}^{S_N^k} = \int_{R_i} \int_{R_j} f_x^{S_N^k}(x) f_b(b-x) dbdx. \tag{2.30}$$

Où la fonction $f_x^{S_N^k}$ correspond à la FDP du signal $\hat{x}_{N+1}^{vflo}$ pour le scénario S_N^k, et f_b est la FDP du bruit total de quantification b_f obtenu dans l'équation (2.29).

La probabilité d'erreur de décision totale est déterminée en considérant l'impact de tous les scénarios possibles sur la sortie de l'opérateur de décision. Par conséquent, la probabilité d'erreur de décision totale, que la sortie du dernier opérateur de décision $\tilde{x}_{N+1}$ corresponde au symbole S_i en arithmétique virgule flottante et S_j en virgule fixe, est une somme de toutes les probabilités d'erreur de décision $P_{i,j}^{S_N^k}$ sous le scénario S_N^k pondérée par la probabilité $P_{S_N^k}$ de réalisation de tous les scénarios possibles S_N^k. Elle est donnée par :

$$\begin{aligned} P_{\substack{i,j \\ i \neq j}} &= \sum_{k=1}^{C^2} P_{S_N^k} P_{i,j}^{S_N^k} \\ &= \sum_{k=1}^{C^2} \int_{R_i} \int_{R_j} P_{S_N^k} f_x^{S_N^k}(x) f_b(b-x) dbdx. \end{aligned} \tag{2.31}$$

Afin de déterminer la probabilité d'erreur de décision $P_{i,j}$ au niveau du dernier opérateur de décision du système considéré, il est nécessaire d'évaluer la probabilité $P_{S_N^k}$ de l'occurrence du scénario S_N^k et la FDP $f_x^{S_N^k}$ du signal $\tilde{x}_{N+1}$ à l'entrée du $N+1^{eme}$ opérateur de décision sous contrainte du scénario S_N^k.

En ce qui concerne la probabilité $P_{S_N^k}$, nous pouvons distinguer le cas où les erreurs de décision sont dépendantes et le cas où elles sont indépendantes. La probabilité $P_{S_N^k}$ est obtenue comme la probabilité conjointe qu'une combinaison des sorties d'un opérateur de décision se produise selon la description du scénario. Par conséquent, la probabilité de l'occurrence du scénario $P_{S_N^k}$ est donnée par :

$$P_{S_N^k} = P((\tilde{x}_1^{vflo} = S_i^1, \tilde{x}_1^{vfix} = S_j^1) \quad et \quad ... \quad et \quad (\tilde{x}_N^{vflo} = S_i^N, \tilde{x}_N^{vfix} = S_j^N)) \tag{2.32}$$

où S_i^r et S_j^r représentent les valeurs prises par la sortie du $r^{ème}$ opérateur de décision respectivement en virgule flottante et en virgule fixe. En d'autres termes, $P_{S_N^k}$ est la probabilité de l'occurrence du $k^{ème}$ scénario.

Dans le cas où les éléments du vecteur $\tilde{X}$ ne sont pas corrélés, la probabilité conjointe est déterminée par :

$$P_{S_N^k} = \prod_{r=1}^{N} P(\tilde{x}_r^{vflo} = S_i^r, \tilde{x}_r^{vfix} = S_j^r) \tag{2.33}$$

Par conséquent, dans ce cas, nous avons tous les éléments nécessaires pour déterminer analytiquement la probabilité d'erreur de décision à la sortie de chaque opérateur de décision du système y compris le dernier opérateur de décision de la cascade.

Par contre, en pratique, nous pouvons rencontrer le cas où les sorties des opérateurs de décision sont corrélées entre elles. La corrélation est déterminée par la topologie du flot du signal et la corrélation entre les vecteurs en entrée utilisés en simulation virgule flottante pour déterminer la FDP du signal. Sous ces conditions, l'impact de chaque sortie de décision sur les autres sorties de décision doit être considéré afin de calculer la probabilité conjointe de l'occurrence des sorties de décision dans le cas du scénario S_N^k.

Soit S_P^k un sous scénario correspondant à un sous ensemble du scénario S_N^k. Par conséquent, les éléments des vecteurs $\tilde{X}_{vflo}$ et $\tilde{X}_{vfix}$ ont une longueur P ($P \leq N$) et ils sont égaux aux premiers P éléments définis dans le vecteur S_N^k. Ainsi, la probabilité conjointe $P_{S_N^k}$ peut être évaluée à partir des probabilités conditionnelles :

$$\begin{aligned}
P_{S_N^k} &= P((\tilde{x}_N^{vflo} = S_i, \tilde{x}_N^{vfix} = S_j)|S_{N-1}^k).P_{S_{N-1}^k} \\
&= P((\tilde{x}_N^{vflo} = S_i, \tilde{x}_N^{vfix} = S_j)|S_{N-1}^k).P((\tilde{x}_{N-1}^{vflo} = S_i, \tilde{x}_{N-1}^{vfix} = S_j)|S_{N-2}^k). \\
&\quad ... \quad P((\tilde{x}_2^{vflo} = S_i, \tilde{x}_2^{vfix} = S_j)|S_1^k).\underbrace{P(\tilde{x}_1^{vflo} = S_i, \tilde{x}_1^{vfix} = S_j)}_{P_{S_1^k}} \\
&= \prod_{r=0}^{N-1} P((\tilde{x}_{N-r}^{vflo} = S_i, \tilde{x}_{N-r}^{vfix} = S_j)|S_{N-r-1}^k).
\end{aligned} \tag{2.34}$$

D'après les équations (2.33) et (2.34), il est clair que le calcul de la probabilité de l'occurrence du scénario S_N^k nécessite la connaissance des probabilités d'erreurs ainsi que les probabilités conjointes des N entrées de décision. Par conséquent, la probabilité de l'erreur $P(\tilde{x}_{vflo} = S_i, \tilde{x}_{vfix} = S_j)$ des sorties de décision doit être calculée *a priori*. Cela veut dire que la probabilité de l'erreur en sortie de chaque opérateur de décision du système doit être calculée dans l'ordre de priorité à partir de l'entrée jusqu'à la sortie du système considéré. Après avoir identifié les opérateurs *non lisses*, il est donc nécessaire d'analyser ces dépendances à partir de l'analyse du graphe flot du système. Lorsque les entrées de décision ne sont pas corrélées, la probabilité conjointe est obtenue par un simple produit entre les différentes probabilités individuelles des scénarios de décision en arithmétique virgule flottante et virgule fixe. D'autres part, dans le cas où les sorties de décision sont corrélées avec d'autres sorties de décision, le calcul de la fonction de probabilité d'erreur de décision du premier opérateur de décision du système se fait par le modèle de l'équation (2.17). La probabilité d'erreur de décision des opérateurs intermédiaires dans le système est déterminée en calculant successivement les probabilités conjointes des erreurs intervenant au sein de l'équation (2.34). Ensuite, les résultats intermédiaires obtenus durant le calcul des différentes probabilités d'erreur de décision sont réutilisés pour le calcul des probabilités des scénarios pour les opérateurs de décision situés en aval.

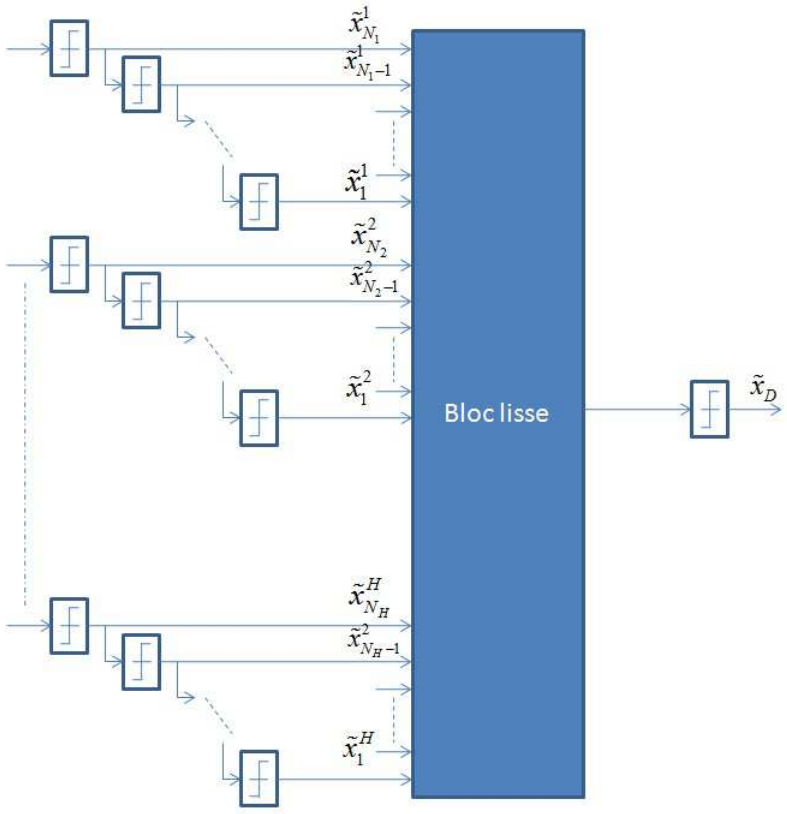

FIGURE 2.14 – Propagation des erreurs non lisses (corrélées ou non corrélées)

Afin de bien illustrer les résultats analytiques établis précédemment, nous considérons le cas générique représenté par le système de la figure 2.14 constitué d'un bloc lisse possédant en entrée H ($H \geq 2$) groupes non lisses. Par ailleurs, nous supposons que le signal de l'un des groupes est indépendant des signaux appartenant aux autres groupes mais qu'il est corrélé avec les signaux du même groupe.

Soit $\tilde{x}_D$ le signal en sortie de l'opérateur de décision situé en sortie du bloc lisse. Dans ce cas, les N entrées de décision sont différenciées en H groupes indépendants de $N_1, N_2, ..., N_H$ opérateurs de décision. N'importe quel i^{eme} opérateur de décision est dépendant du $i-1^{eme}$ opérateur de décision du même groupe.

Afin d'être en mesure de déterminer analytiquement la probabilité de l'occurrence de tous les scénarios possibles résultant des $N_1 + N_2 + ... + N_H$ entrées, les différents scénarios sont définis par toutes les entrées possibles des opérateurs *non lisses* de décision. Comme les H groupes des signaux ne sont pas corrélés, la probabilité de l'occurrence de ces scénarios peut être obtenue par le produit de la probabilité de l'occurrence des scénarios de chaque groupe. Par conséquent, la probabilité totale des scénarios en entrée qui considère les entrées des opérateurs *non lisses* de décision peut être calculée par :

$$P_{S^k_{N_1+N_2+...+N_H}} = \prod_{r=1}^{H} P_{S^k_{N_r}} \tag{2.35}$$

où $P_{S^k_{N_r}}$, donnée par l'équation (2.34) désigne la probabilité de l'occurrence des scénarios du r^{eme} groupe.

2.3.3 Analyse de la complexité du modèle analytique

Le calcul d'une probabilité conditionnelle est similaire à celui de la probabilité $P_{i,j}$ du premier opérateur de décision. Le coût calculatoire nécessaire pour évaluer la probabilité conjointe peut

être mesuré en termes de nombre d'unités d'évaluation numérique de l'intégrale exprimant les $P_{i,j}$ pour un scénario donné. Ce nombre d'unités d'évaluation dépend bien entendu du nombre de scénarios considérés. Le nombre de fois de l'évaluation dépend du nombre des scénarios. Ce dernier augmente exponentiellement avec le nombre des entrées de décision *non lisses* qui doivent être prises en considération. Dans tous les cas (corrélation ou non corrélation), il est inévitable d'évaluer ces nombreuses intégrales pour un nombre donné de combinaisons possibles d'erreurs. Supposons qu'il y ait M_k symboles de décision à la sortie du $k^{ème}$ opérateur de décision, ils contribuent à M_k combinaisons de scénario. En considérant toutes les combinaisons possibles des sorties des opérateurs de décision, il y a au total $C = \prod_{k=1}^{N} M_k$ combinaisons possibles. Par conséquent, la complexité du calcul des probabilités d'erreur à la sortie du $(N+1)^{ème}$ opérateur de décision (dépendant des N précédents opérateurs de décision) est de l'ordre de de $O(C^2)$, où C possède une complexité exponentielle.

Afin d'illustrer l'étude mathématique analytique de la section précédente, nous considérons le cas d'une application issue du domaine des communications numériques : le décodage sphérique. Plus particulièrement, nous avons choisi de travailler sur l'algorithme SSFE (*Selective Spanning with Fast Enumeration*) car il représente un très bon cas d'application d'un système de communication numérique constitué d'une cascade d'opérateurs de décision. Cette étude est présentée dans la prochaine section.

2.4 Application du modèle à l'algorithme SSFE

Les systèmes de communications numériques sans fils avec plusieurs entrées et plusieurs sorties MIMO (*Multiple-Input Multiple-Output*) utilisent plusieurs antennes dans les deux extrémités du canal sans fil. En appliquant le multiplexage spatial (transmission de multiples flux de données concurrentes dans une même bande de fréquence à l'aide de plusieurs antennes), les systèmes sans fil MIMO offrent une augmentation de l'efficacité spectrale par rapport aux systèmes mono-antenne. Par conséquent, les systèmes sans fil MIMO sont en mesure de répondre à la demande des débits plus élevés sans augmenter la bande passante. Ils sont donc devenus la technologie de base pour toutes les prochaines normes de communication sans fils, telles que IEEE 802.11n, WiMAX, 3GPP LTE et 3GPP2 UMB.

Pour couvrir la vaste gamme d'applications, les ordinateurs futurs devront supporter simultanément une grande variété de normes de communication sans fil. Avec le nombre croissant d'interfaces "air" qui doit être pris en charge, les implémentations traditionnelles, basées sur l'intégration de plusieurs radios spécifiques, deviennent peu rentables. Cela souligne la nécessité de solutions multi-standards flexibles. Dans cet esprit, le détecteur MIMO sépare spatialement les flux de données multiplexées au niveau du côté du récepteur. Pour la mise en œuvre du détecteur, une large gamme d'algorithmes de détection sont disponibles dans la littérature.

Le choix d'un algorithme spécifique influence significativement le rendement ainsi que la complexité de la solution matérielle résultante. Parmi les algorithmes les plus utilisés, les récepteurs linéaires à sortie ferme (*hard* en anglais) possèdent une faible complexité mais souffrent d'une performance limitée en termes de BER (*Bit Error Rate*). En revanche, le choix d'un récepteur à sortie souple (*soft* en anglais) basé sur une détection à maximum de vraisemblance (MV) offre une performance maximale mais au prix d'une grande complexité entraînant une consommation d'énergie importante. Afin de satisfaire aux exigences énergétiques des appareils portables du futur, la conception de récepteurs MIMO et leurs implémentations optimisées sont donc devenues un défi majeur.

Plusieurs techniques de détection MIMO ont été proposées telles que la détection linéaire, la détection SIC (*Successive Interference Cancelation*) ou bien encore les algorithmes de type *K-Best*. Toutefois, ces détecteurs MIMO sont généralement liés à un schéma de modulation et ne sont pas évolutifs lorsque l'on souhaite changer le nombre d'antennes [66]. En outre, leur consommation d'énergie est encore assez élevée. Par conséquent, ils ne sont généralement pas

adaptés pour les systèmes multi-standards économes en énergie.

C'est pourquoi, nous nous intéressons dans ce chapitre à l'optimisation de l'algorithme SSFE basé sur un *spanning* sélectif avec énumération rapide ce qui permet de résoudre les problèmes de la détection MV à l'aide d'une décomposition orthogonale triangulaire de la matrice H de canal. Comme nous l'avons précisé dans le premier chapitre, l'arithmétique en virgule fixe répond aux besoins d'implémentation en termes d'une faible complexité et d'une consommation d'énergie réduite par rapport à la représentation équivalente en virgule flottante.

En contre partie, la précision en arithmétique virgule fixe doit être maitrisée et évaluée afin de garantir l'intégrité de l'application conçue. Dans cette section, nous présentons tout d'abord le fonctionnement de l'algorithme SSFE puis nous proposerons un modèle analytique afin d'évaluer la précision en sortie de chaque antenne de détection à la réception en se basant sur le modèle analytique présenté précédemment dans le cadre d'une cascade d'opérateurs *non lisses* de décision.

2.4.1 Modèle du système MIMO

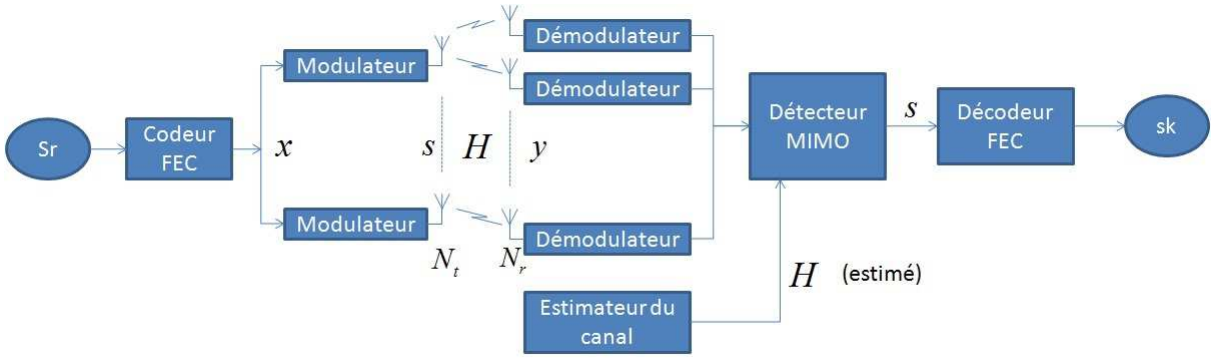

FIGURE 2.15 – Propagation des erreurs non lisses (corrélées et non corrélées)

Le modèle du système utilisé est illustré par la figure 2.15. Pour une raison d'exhaustivité, les blocs de correction d'erreur avant FEC (Forward Error Correction) sont montrés dans la figure. Le nombre d'antennes est égal à N_t et N_r respectivement à l'émission et à la réception. Pour une modulation de type W-QAM, un symbole représente l'un de $W = 2^g$ points de constellation. L'émetteur transmet un vecteur binaire x de taille $gN_t \times 1$ au sein de chaque vecteur symbole s de taille $N_t \times 1$. La transmission de chaque vecteur s à travers le canal MIMO à évanouissement non sélectif en fréquence peut être modélisée par $y = H.s + n$, où y représente le vecteur symbole reçu de taille $N_r \times 1$, H caractérise la matrice canal de taille $N_t \times N_r$ et n est le vecteur bruit.

La tâche du détecteur MIMO consiste à retrouver le vecteur symbole s transmis par le transmetteur.

Les détecteurs MIMO à sortie souple fournissent non seulement le vecteur de symboles le plus probable s (comme le font les détecteurs à sortie *hard*), mais aussi le rapport de vraisemblance logarithmique LLR (*Log-Likelihood Ratio*) qui correspond au rapport entre les vraisemblances de la valeur mesurée à l'entrée du récepteur sous les deux conditions possibles d'émission d'un bit s égal à 0 ou 1.

Les décodeurs modernes, tels que les turbo-décodeurs et les décodeurs LDPC, qui feront l'objet du dernier chapitre de cette thèse, constituent un élément critique essentiel des futurs systèmes de transmission et nécessitent l'utilisation d'entrées souples (ou *soft*) afin d'obtenir des performances optimales en termes de BER. De tels décodeurs comportent deux éléments principaux : un générateur de liste et un générateur LLR.

Le générateur de liste détermine une liste L des vecteurs s les plus probables. Parmi les techniques classiquement utilisées pour effectuer cette opération, le détecteur à maximum de vraisemblance offre de meilleures performances en termes de BER mais il possède une complexité

la plus importante. Dans cette section, nous exploitons la technique de réception de type SSFE pour effectuer la génération de liste dans le contexte des transmissions de type MIMO.

2.4.2 Présentation de l'algorithme

Contrairement à l'algorithme *K-best* traditionnellement utilisé, l'algorithme SSFE possède une structure de type flux de données et ne consomme pas beaucoup de mémoire pour les opérations. De plus, le SSFE est basé sur des opérateurs simples et lisses telles que l'addition, le décalage, la soustraction et également la cascade d'opérateurs de décision, ce qui réduit la complexité de l'implémentation. Par ailleurs, il possède l'avantage d'être paramétrable ce qui permet de régler ses paramètres en fonction du compromis complexité/performance ciblé.

Pour la détection MV, le détecteur prend ses décisions suivant la règle :

$$\hat{s} = \underset{s \in \Omega^{Nt}}{\arg\min} \|y - Hs\|^2 \tag{2.36}$$

avec Ω^{N_t} désignant l'ensemble contenant toutes les réalisation possibles du vecteur signal s de taille $N_t \times 1$. Pour la détection MV-proche (*near-ML*), seulement un nombre limité de vecteur s sont considérés.

Une instance de l'algorithme SSFE est caractérisée uniquement par un vecteur scalaire $m = [m_1, ..., m_{N_t}], \quad m_i \geq W$. Les entrées dans ce vecteur spécifient le nombre des symboles scalaires s_i qui sont pris en compte au niveau de l'antenne N_i. Il est important de noter qu'avec le paramètre m, un compromis complexité/performance est sélectionné. Le calcul de s peut être visualisé avec l'arbre de *spanning*. Commençant au niveau $i = N_t$, le SSFE traverse chaque nœud au niveau $i + 1$ jusqu'à m_i nœuds. Un exemple d'un arbre avec $m = [1, 1, 2, 4]$ est illustré dans la figure 2.16.

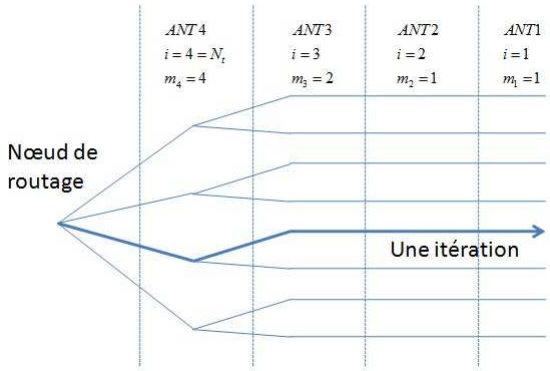

FIGURE 2.16 – Exemple d'un arbre de spanning SSFE avec $m = [1, 1, 2, 4]$

Le premier nœud racine est noté par $T_{N_t+1} = 0$. En commençant du niveau $i = N_t$, la distance euclidienne partielle DEP d'un vecteur symbole $s^i = [s_i, s_{i+1}, ..., s_{N_t}]$ est donnée par :

$$T_i(s^i) = T_{i+1}(s^{i+1}) + \left\| e_i(s^i) \right\|^2 \tag{2.37}$$

avec $\|e_i(s^i)\|^2$ décrit l'incrémentation de la DEP. L'algorithme SSFE doit sélectionner un ensemble de $s^i = [s_i, s_{i+1}, ..., s_{N_t}]$ tel que le pas d'incrémentation de la DEP $\|e_i(s^i)\|^2$ soit minimisé. En considérant une décomposition *a priori* QR triangulaire orthogonale de la matrice canal H ($H = QR$), le pas pour incrémenter la DEP peut être calculé comme suit :

$$\left\|e_i(s^i)\right\|^2 = \left\| \underbrace{\hat{y}_i - \sum_{j=i+1}^{N_t} R_{ij}s_j}_{b_{i+1}(s^{i+1})} - R_{ii}s_i \right\|^2 . \tag{2.38}$$

Les termes R_{ij} sont les coefficients de la matrice R. Comme la minimisation de $\|e_i(s^i)\|^2$ est équivalente à la minimisation de $\|e_i(s^i)/R_{ii}\|^2$, l'équation précédente peut être transformée (si R_{ii} n'est pas nul) selon :

$$\left\|e_i(s^i)/R_{ii}\right\|^2 = \left\| \underbrace{b_{i+1}(s^{i+1})/R_{ii}}_{\epsilon_i} - s_i \right\|^2 = \|\epsilon_i - s_i\|^2 . \tag{2.39}$$

Nous pouvons également définir le vecteur $y^{'}$ défini par :

$$\|y - H.s\|^2 = c + \left\|y^{'} - R.s\right\|^2 , y^{'} = Q^t.y \tag{2.40}$$

avec c est une constante.

La tâche du SSFE est de sélectionner un ensemble des points de constellation les plus proches autour de ϵ_i. Ceci est fait essentiellement en minimisant $\|e_i(s^i)/R_{ii}\|^2$ dans l'équation (2.39). Lorsque $m_i = 1$, le point de constellation le plus proche de ϵ_i est $p_1 = D(\epsilon_i)$, où D désigne l'opérateur de décision. Lorsque $m_i > 1$, plusieurs constellations peuvent être énumérées en se basant sur le vecteur $d = \epsilon_i - D(\epsilon_i)$. Le principe fondamental de cet algorithme consiste donc à augmenter itérativement l'ensemble autour de ϵ_i en appliquant des approximations heuristiques. Un exemple d'une énumération de 8-point est montrée dans la figure 2.17. Le premier point et le dernier point sont notés respectivement par les chiffres 1 et 8. Nous pouvons énumérer plusieurs points de la même manière. L'énumération rapide (EN) possède des avantages clairs avec la modulation PSK tel que l'énumération implémentée dans [66]. Les performances de cet algorithme en termes de BER sont présentées dans [66] ainsi que des comparaisons avec d'autres algorithmes concurrents .

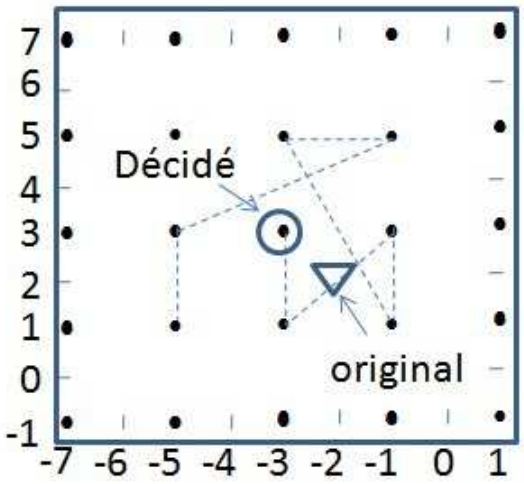

FIGURE 2.17 – Exemple d'une énumération rapide (ER) d'une constellation à 8 points

A la réception, l'algorithme SSFE traite le vecteur $y^{'}$ pour effectuer des opérations arithmétiques lisses et des opérations de décisions en cascade. A titre d'exemple, la figure 2.18 représente,

dans le cas de 4 antennes, le traitement effectué à la réception par l'algorithme SSFE sur le vecteur $y^{'}$.

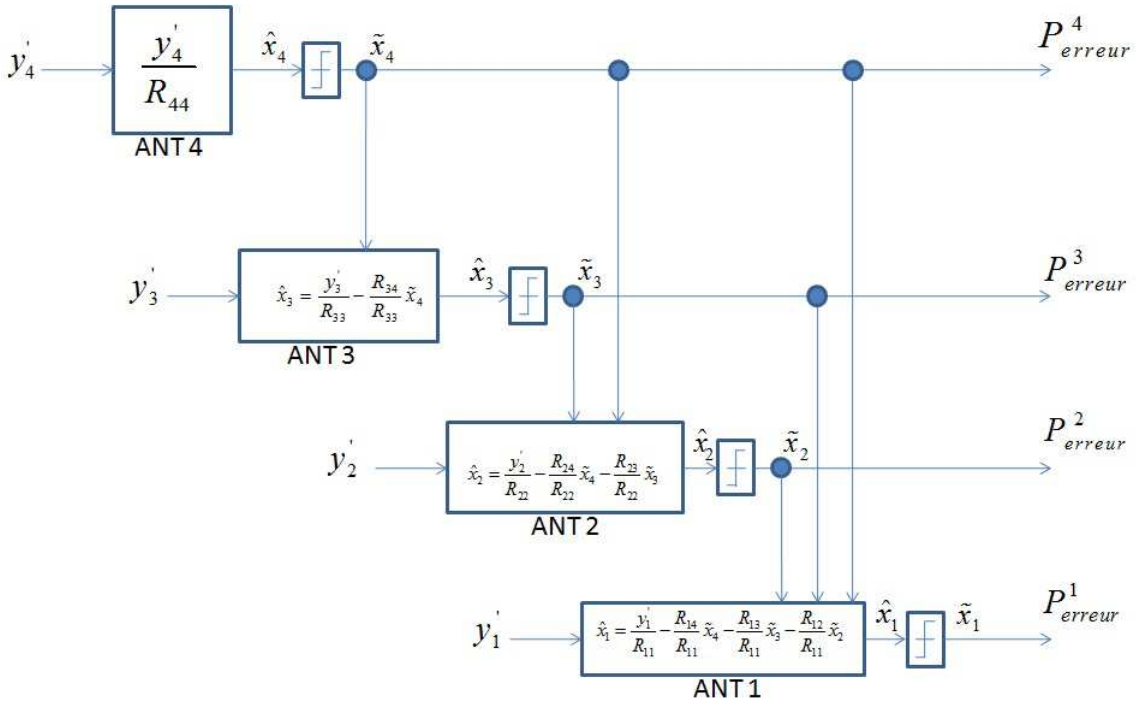

FIGURE 2.18 – Cascade d'opérateurs de décisions : cas de l'algorithme SSFE

2.4.3 Application du modèle analytique proposé

Dans cette section, nous proposons d'appliquer le modèle analytique présenté précédemment au cas particulier de l'algorithme SSFE à 4 antennes, ceci afin d'évaluer la précision d'une cascade d'opérateurs de décisions non lisses en tenant compte de la propagation de l'erreur causée par le bruit de quantification du format virgule fixe à travers la cascade existante dans le système de l'antenne 4 jusqu'à l'antenne 1. Afin de pouvoir utiliser le modèle de l'équation (3.4), nous devons modéliser cette propagation du bruit de quantification à l'entrée de chaque opérateur de la cascade. Nous présenterons deux approches de modélisation et de prise en compte de la propagation de l'erreur causée par le bruit de quantification. La première approche consiste à tenir compte de la propagation de l'erreur causée par le bruit de quantification dans la FDP du bruit de quantification à l'entrée de l'opérateur de décision courant. La deuxième approche consiste à modéliser cette propagation de l'erreur dans la FDP du signal présent à l'entrée de chaque opérateur de décision de la cascade.

2.4.4 Première approche

Afin d'illustrer cette approche simplement, nous présentons tout d'abord l'étude en considérant le cas d'une modulation BPSK où la FDP du signal reçu sur chaque antenne est composée par deux gaussiennes centrées sur -1 et 1 avec une variance σ_{ch}^2. Soit $f_x^4(x)$ la FDP du signal à l'entrée de l'opérateur de décision de l'antenne 4 qui est le premier opérateur de la cascade de l'algorithme SSFE. Dans le cas où les signaux de la constellation BPSK sont équiprobables, cette FDP est donnée par :

$$f_X^4(x) = \frac{1}{2.\sqrt{2\pi}.\sigma_{ch}} \left(\exp(-\frac{(x-1)^2}{2\sigma_{ch}^2}) + \exp(-\frac{(x+1)^2}{2\sigma_{ch}^2}) \right). \tag{2.41}$$

Nous considérons également que le bruit de quantification b à l'entrée de chaque opérateur de décision est un bruit blanc gaussien de moyenne nulle dont la FDP est :

$$f_B^4(x) = \frac{1}{\sqrt{2\pi}.\sigma_b}.\exp(-\frac{(x)^2}{2\sigma_b^2}). \tag{2.42}$$

Dans ce cas, il y a deux symboles -1 et 1 dont les régions de décision correspondent respectivement aux intervalles $]-\infty, 0]$ et $[0, +\infty[$. Pour l'antenne 4, il existe deux probabilités d'erreur de décision $P_{-1,1}^4$ et $P_{1,-1}^4$ qui sont déterminées analytiquement par application directe de l'équation (2.17) :

$$\begin{aligned} P_{-1,1}^4 &= \int_{-\infty}^{0}\int_{0}^{+\infty} f_X^4(x) f_B^4(b-x) dbdx \\ P_{1,-1}^4 &= \int_{0}^{+\infty}\int_{-\infty}^{0} f_X^4(x) f_B^4(b-x) dbdx. \end{aligned} \tag{2.43}$$

Et la probabilité d'erreur totale est donnée par :

$$P_{erreur}^4 = P_{-1,1}^4 + P_{1,-1}^4. \tag{2.44}$$

Antenne 3 Dans le cas de l'opérateur de décision de l'antenne 3, l'entrée $\hat{x}_3$ dépend non seulement du signal reçu $y_3^{'}$ mais aussi de la sortie $\tilde{x}_4$ de l'opérateur de décision qui lui précède dans l'antenne 4 :

$$\hat{x}_3 = \frac{y_3^{'}}{R_{33}} - \frac{R_{34}}{R_{33}}.\tilde{x}_4. \tag{2.45}$$

Dépendamment de la valeur décidé $\tilde{x}_4$ qui est soit -1 ou 1 dans le cas BPSK, $\hat{x}_3$ peut prendre les valeurs suivantes :

$$\begin{aligned} \hat{x}_{3_{|\tilde{x}_4=1}} &= \frac{y_3^{'}}{R_{33}} - \frac{R_{34}}{R_{33}} \\ \hat{x}_{3_{|\tilde{x}_4=-1}} &= \frac{y_3^{'}}{R_{33}} + \frac{R_{34}}{R_{33}}. \end{aligned} \tag{2.46}$$

Par conséquent lorsqu'une erreur se produit au niveau du bloc de décision de l'antenne 4, elle se propage probablement jusqu'aux prochaines antennes 3, 2 et 1. La FDP du signal à l'entrée de l'opérateur de décision de l'antenne 3 s'écrit dans ce cas comme suit :

$$f_x^3(x) = p(\tilde{x}_4 = 1).f_{\hat{x}_{3_{|\tilde{x}_4=1}}}^3(x) + p(\tilde{x}_4 = -1).f_{\hat{x}_{3_{|\tilde{x}_4=-1}}}^3(x). \tag{2.47}$$

Les fonctions de densité de probabilité $f_{\hat{x}_{3_{|\tilde{x}_4=1}}}^3(x)$ et $f_{\hat{x}_{3_{|\tilde{x}_4=-1}}}^3(x)$ respectivement des signaux $\hat{x}_{3_{|\tilde{x}_4=1}}$ et $\hat{x}_{3_{|\tilde{x}_4=-1}}$ sont égales à la FDP du signal reçu au niveau de l'antenne 3 $(y_3^{'}/R_{33})$ dont la moyenne est décalée respectivement de $-\frac{R_{34}}{R_{33}}$ et $\frac{R_{34}}{R_{33}}$.

Dans le cas où $(y_3^{'}/R_{33})$ est un signal BPSK possédant une FDP composée de deux gaussiennes équiprobables centrées sur -1 et 1, la FDP du signal $\hat{x}_3$ à l'entrée de l'opérateur de décision s'écrit comme suit :

$$\begin{aligned} f_x^3(x) &= p(\tilde{x}_4 = 1)\frac{1}{2.\sqrt{2\pi}.\sigma_{ch}}\left(\exp(-\frac{(x-1+\frac{R_{34}}{R_{33}})^2}{2\sigma_{ch}^2}) + \exp(-\frac{(x+1+\frac{R_{34}}{R_{33}})^2}{2\sigma_{ch}^2})\right) \\ &+ p(\tilde{x}_4 = -1)\frac{1}{2.\sqrt{2\pi}.\sigma_{ch}}\left(\exp(-\frac{(x-1-\frac{R_{34}}{R_{33}})^2}{2\sigma_{ch}^2}) + \exp(-\frac{(x+1-\frac{R_{34}}{R_{33}})^2}{2\sigma_{ch}^2})\right). \end{aligned} \tag{2.48}$$

Afin de déterminer analytiquement la probabilité d'erreur de décision au niveau de l'antenne n°3, il est nécessaire de déterminer la FDP du bruit à l'entrée de l'opérateur de décision. Ce bruit n'est pas seulement composé par le bruit de quantification dû à l'arithmétique en virgule fixe, mais aussi par le bruit causé par une erreur de décision produite au niveau de l'antenne précédente n°4 et propagée à l'antenne actuelle n°3. En effet, lorsqu'une erreur est produite au niveau de l'antenne précédente n°4 tel que le scénario correspondant S_n^k est une décision du symbole S_j en arithmétique virgule fixe mais elle est le symbole S_i en arithmétique virgule flottante, le bruit de quantification au niveau de l'antenne n°3 sera décalé par $d_{ij}\frac{R_{34}}{R_{33}}$ où $d_{ij} = S_i - S_j$ définie par l'équation (2.21), représente la différence entre les symboles de la constellation impliqués dans le scénario de l'erreur produite au niveau du bloc de décision précédent. Lorsque $i \neq j$, il s'agit bien d'une erreur de décision, tandis que lorsque $i = j$ ($d_{ij} = 0$), il s'agit du cas où il n y avait pas d'erreur de décision. Ce raisonnement englobe les deux cas ($i \neq j$ et $i = j$).

Par conséquent, il convient de préciser, que selon cette approche, le bruit à l'entrée de l'opérateur de décision en cours est une somme pondérée de plusieurs bruits. Chaque bruit de la somme caractérise un scénario produit au niveau du précédent opérateur de décision multiplié par la probabilité d'occurrence de ce scénario. Dans le cas d'une constellation BPSK, le bruit à l'entrée de l'opérateur de décision de l'antenne n°3 est une somme pondérée de trois composantes. La première composante est le bruit de quantification de la conversion en virgule fixe au niveau de l'antenne n°3, que nous supposons blanc gaussien et centré, pondéré par probabilité de bonne décision au niveau du bloc de décision de l'antenne précédente ($1 - P^4_{erreur}$). La deuxième composante caractérise le cas où il y avait une erreur de décision de type $P^4_{-1,1}$ au niveau de l'antenne n°4. Cette composante n'est que le bruit de quantification au niveau de l'antenne n°3 décalée par $d_{-1,1}.\frac{R_{34}}{R_{33}} = -2.\frac{R_{34}}{R_{33}}$ pondéré par la probabilité de l'occurrence de ce scénario qui est $P^4_{-1,1}$. La troisième composante est l'autre cas de l'erreur de décision au niveau de l'opérateur de décision de l'antenne n°4 dont la probabilité est $P^4_{1,-1}$. De même, cette composante est le bruit de quantification de l'antenne n°3 décalé de $2.\frac{R_{34}}{R_{33}}$. Par conséquent la FDP du bruit présent au niveau de l'opérateur de décision de l'antenne n°3 est :

$$f^3_B(b-x) = P^4_{-1,1}.f^3_{B,-2.\frac{R_{34}}{R_{33}}}(b-x) + P^4_{1,-1}.f^3_{B,2.\frac{R_{34}}{R_{33}}}(b-x) + (1-P^4_{erreur}).f^3_{B,0}(b-x). \quad (2.49)$$

Dans le cas d'un bruit blanc gaussien, cette FDP s'écrit par :

$$\begin{aligned} f^3_B(b-x) &= P^4_{-1,1}.\frac{1}{\sqrt{2\pi}.\sigma_b}.\exp(-\frac{(b-x-2.\frac{R_{34}}{R_{33}})^2}{2\sigma_b^2}) + P^4_{1,-1}.\frac{1}{\sqrt{2\pi}.\sigma_b}.\exp(-\frac{(b-x+2.\frac{R_{34}}{R_{33}})^2}{2\sigma_b^2}) \\ &+ (1-P^4_{erreur}).\frac{1}{\sqrt{2\pi}.\sigma_b}.\exp(-\frac{(b-x)^2}{2\sigma_b^2}). \end{aligned} \quad (2.50)$$

Après avoir déterminé les fonctions densité de probabilité du signal $\hat{x}_3$ et du bruit b_3 à l'entrée de l'opérateur de décision de l'antenne n°3, nous pouvons déterminer les probabilités d'erreurs de décision $P^3_{i,j}$ par application directe de l'équation (2.17) :

$$P^3_{i,j} = \int_{R_i}\int_{R_j} f^3_X(x) f^3_B(b-x) dbdx. \quad (2.51)$$

Dans le cas BPSK, les probabilités d'erreurs de décision sont :

$$\begin{aligned} P^3_{-1,1} &= \int_{-\infty}^{0}\int_{0}^{+\infty} f^3_X(x) f^3_B(b-x) dbdx \\ P^3_{1,-1} &= \int_{0}^{+\infty}\int_{-\infty}^{0} f^3_X(x) f^3_B(b-x) dbdx. \end{aligned} \quad (2.52)$$

Nous avons calculé la probabilité d'erreur de décision pour l'opérateur de décision de l'antenne n°3 qui constitue une cascade avec le précédent opérateur de décision de l'antenne n°4. Cette technique prend en compte la propagation du bruit de l'erreur produite au niveau de l'antenne n°4 jusqu'à l'antenne actuelle n°3. Cette propagation de l'erreur est modélisée dans la FDP du bruit présent à l'entrée de l'opérateur de décision de l'antenne n°3 en considérant tous les scénarios possibles produits au niveau du bloc de décision de l'antenne n°4.

La figure 2.19 montre la probabilité d'erreur de décision au niveau de l'opérateur de décision de l'antenne 3 obtenue à partir de cette approche analytique ainsi que celle obtenue par simulation pour différentes valeurs du rapport entre la puissance de signal par élément binaire et celle du bruit.

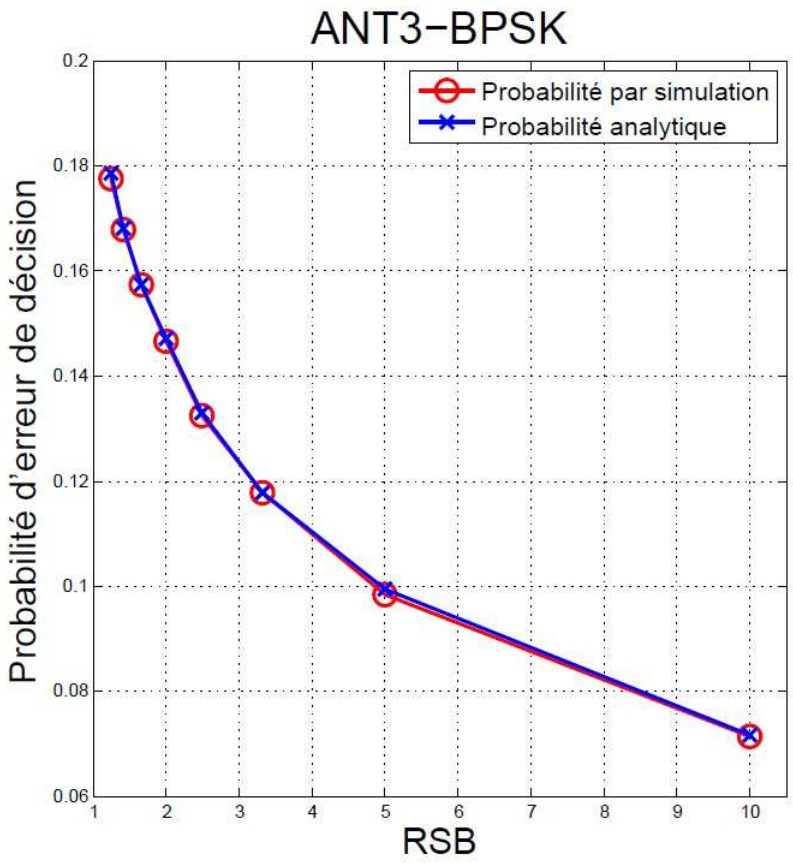

FIGURE 2.19 – Probabilité d'erreur de décision totale en fonction du rapport signal à bruit

Nous remarquons que la probabilité d'erreur de décision calculée analytiquement pour l'opérateur de décision de l'antenne n°3 est conforme à la probabilité d'erreur de décision calculée par la technique de simulation virgule fixe et virgule flottante de l'algorithme SSFE dans le cas d'une constellation BPSK. L'erreur maximale entre la probabilité d'erreur estimée analytiquement et la probabilité estimée par simulation est de l'ordre de 2%.

Il convient de préciser par ailleurs que le temps d'exécution pris par la simulation pour estimer cette probabilité d'erreur de décision est largement supérieur aux temps pris pour évaluer numériquement la probabilité analytique. Nous présenterons ultérieurement une étude comparative entre les temps d'exécution obtenus par la méthode de simulation et par le modèle analytique afin de montrer l'intérêt de notre contribution pour l'évaluation des applications contenant des bloc lisses et des cascades d'opérateurs *non lisses* de décision.

Cas générique Dans le cas où les signaux reçus sont de type M-QAM avec une équiprobabilité entre les différents points de constellation, nous suivons également la même stratégie que celle

utilisée dans le cas BPSK afin de déterminer la fonction de densité de probabilité du signal présent à l'entrée de l'opérateur de décision de l'antenne n°3 ainsi que celle du bruit présent en entrée de l'opérateur. Nous appliquerons ensuite l'intégrale de l'équation (2.17) pour obtenir les différentes probabilités d'erreur de décision $P^3_{i,j}$.

Afin d'évaluer de façon analytique les performances de décision de l'antenne n°3, il est nécessaire de calculer en premier lieu les FDP $f^3_x(x)$ et $f_B(b-x)$. Supposons que la sortie de l'opérateur de décision de l'antenne n°4 soit $\tilde{x}_4 = S_i$, alors l'entrée de l'opérateur courant de décision est donnée d'après (2.45) par :

$$\hat{x}_{3_{|\tilde{x}_4=S_i}} = \frac{y_3^{'}}{R_{33}} - \frac{R_{34}}{R_{33}}.S_i. \tag{2.53}$$

Par conséquent, la FDP de $\hat{x}_3$ est définie par :

$$f^3_x = \sum_{t=1}^{M} P(\tilde{x}_4 = S_t) f_{\hat{x}_{3_{|\tilde{x}_4=S_i}}} \tag{2.54}$$

où $f_{\hat{x}_{3_{|\tilde{x}_4=S_i}}}$ désigne la FDP du signal $\hat{x}_{3_{|\tilde{x}_4=S_i}}$ qui correspond encore, d'après (2.53), à la FDP du signal $\frac{y_3^{'}}{R_{33}}$ décalée par $\frac{R_{34}}{R_{33}}.S_i$.

Après cette étape, nous devons déterminer la FDP du bruit b_3 à l'entrée de l'opérateur de décision de l'antenne n°3 en tenant compte de la propagation jusqu'à l'antenne n°3 d'une erreur de décision due à l'arithmétique virgule fixe et en considérant tous les scénarios possibles d'erreurs produites dans le bloc de décision précédent de l'antenne n°4. Par conséquent la FDP $f^3_B(x)$ s'écrit par :

$$f^3_b(x) = \sum_{i=1}^{M}\sum_{j=1}^{M} P^4_{i,j}.f^3_{B|d_{ij}}(x) \tag{2.55}$$

où $f_{B|d_{ij}}$ désigne la FDP de la composante du bruit présent en entrée de l'opérateur de décision et correspondant à un scénario d'erreur $P^4_{i,j}$ au niveau de l'opérateur de décision de l'antenne n°4. Cette FDP est la FDP $f^3_{B|0}(b-x)$ du bruit de quantification causé par la conversion du signal $\hat{x}_3$ en entrée de l'opérateur de décision en virgule fixe décalée par $\frac{R_{34}}{R_{33}}.d_{ij} = \frac{R_{34}}{R_{33}}.(S_i - S_j)$.

Arrivant à ce stade, nous avons tous les éléments nécessaires pour évaluer la probabilité d'erreur de décision causée par la conversion en format virgule fixe et la propagation des erreurs de quantification pour l'antenne n°3 dans le cas d'une constellation de type M-QAM.

Antenne 2 : Considérons maintenant l'étape suivante de l'architecture du récepteur SSFE, c'est à dire celle qui consiste à évaluer la précision de l'opérateur de décision situé au niveau de l'antenne n°2. Comme dans le cas de l'antenne n°3, nous commencerons par une étude dans le cas d'une constellation de type BPSK pour mettre en évidence les mécanismes intervenant dans la propagation du bruit d'un opérateur à l'autre de la cascade et nous présenterons le cas générique pour une constellation de type QAM.

Dans le cas de l'antenne n°2, l'opérateur de décision constitue une cascade avec les deux opérateurs de décision correspondant aux antennes n°2 et n°3 situées en amont. Par conséquent l'entrée $\hat{x}_2$ de l'opérateur de décision de l'antenne n°2 ne dépend pas seulement du signal reçu $y_2^{'}$ mais aussi des sorties $\tilde{x}_3$ et $\tilde{x}_4$ des opérateurs de décision respectivement des antennes n°3 et n°4 :

$$\hat{x}_2 = \frac{y_2^{'}}{R_{22}} - \frac{R_{24}}{R_{22}}\tilde{x}_4 - \frac{R_{23}}{R_{22}}\tilde{x}_3. \tag{2.56}$$

Afin de déterminer les probabilités d'erreur de décision $P^2_{i,j}$ (lorsque i est égal à j, on obtient alors les probabilités de bonne décision) au niveau de l'opérateur de décision de l'antenne n°2 et appliquer l'équation (2.17), il faut déterminer la FDP du signal $\hat{x}_2$ présent en entrée de l'opérateur de décision $f^{(2)}_x(x)$ mais également la FDP $f^{(2)}_B(x)$ du bruit b_2 présent à cette même entrée. Selon cette approche, nous modélisons la propagation des erreurs dans la FDP du bruit en entrée de

l'opérateur de décision. A partir de la connaissance de ces FDP, il est possible d'analyser le phénomène de propagation des erreurs à partir de l'analyse de la FDP du bruit présent en entrée de l'opérateur de décision selon l'approche précédemment présentée. Les erreurs survenues au sein des opérateurs de décision des antennes n°3 et 4, situées en amont, vont donc se propager au sein du système et ceci jusqu'à l'antenne n°2. Nous commençons par déterminer la FDP du signal en entrée de l'opérateur de décision. Dans le cas d'une modulation BPSK prenant ses valeurs dans l'alphabet $\{-1,+1\}$, le signal $\hat{x}_2$ peut prendre quatre valeurs dépendamment des sorties $\tilde{x}_3$ et $\tilde{x}_4$ des opérateurs de décision des antennes n°3 et 4 respectivement. Ce signal est donné par :

$$\begin{aligned}
\hat{x}_{2_{|\tilde{x}_4=1,\tilde{x}_3=1}} &= \frac{y_2^{'}}{R_{22}} - \frac{R_{24}}{R_{22}} - \frac{R_{23}}{R_{22}} \\
\hat{x}_{2_{|\tilde{x}_4=-1,\tilde{x}_3=-1}} &= \frac{y_2^{'}}{R_{22}} + \frac{R_{24}}{R_{22}} + \frac{R_{23}}{R_{22}} \\
\hat{x}_{2_{|\tilde{x}_4=1,\tilde{x}_3=-1}} &= \frac{y_2^{'}}{R_{22}} - \frac{R_{24}}{R_{22}} + \frac{R_{23}}{R_{22}} \\
\hat{x}_{2_{|\tilde{x}_4=-1,\tilde{x}_3=1}} &= \frac{y_2^{'}}{R_{22}} + \frac{R_{24}}{R_{22}} - \frac{R_{23}}{R_{22}}.
\end{aligned} \tag{2.57}$$

Par conséquent, la FDP du signal $\hat{x}_2$ à l'entrée de l'opérateur de décision de l'antenne n°2 est donnée par :

$$\begin{aligned}
f_x^{(2)}(x) &= p(\tilde{x}_4=1,\tilde{x}_3=1).f^2_{\hat{x}_{2_{|\tilde{x}_4=1,\tilde{x}_3=1}}}(x) + p(\tilde{x}_4=-1,\tilde{x}_3=-1).f^2_{\hat{x}_{2_{|\tilde{x}_4=-1,\tilde{x}_3=-1}}}(x) \\
&+ p(\tilde{x}_4=-1,\tilde{x}_3=1).f^2_{\hat{x}_{2_{|\tilde{x}_4=-1,\tilde{x}_3=1}}}(x) + p(\tilde{x}_4=1,\tilde{x}_3=-1).f^2_{\hat{x}_{2_{|\tilde{x}_4=1,\tilde{x}_3=-1}}}(x)
\end{aligned} \tag{2.58}$$

où $f_{\hat{x}_{2_{|\tilde{x}_4=i,\tilde{x}_3=j}}}$ désigne la FDP du signal reçu sur l'antenne n°2 ($\frac{y_2^{'}}{R_{22}}$ décalé par $-S_i.\frac{R_{24}}{R_{22}} - S_j.\frac{R_{23}}{R_{22}}$. Par conséquent, il est nécessaire de calculer la probabilité $p(\tilde{x}_4=S_i,\tilde{x}_3=S_j)$. Celle ci est donnée par :

$$p(\tilde{x}_4=S_i,\tilde{x}_3=S_j) = p(\tilde{x}_3=S_j|\tilde{x}_4=S_i).p(\tilde{x}_4=S_i). \tag{2.59}$$

Or le signal $\hat{x}_{3|\tilde{x}_4=S_i}$ présent en entrée de l'opérateur de décision de l'antenne n°3 sous condition que la sortie de l'opérateur de décision de l'antenne n°4 soit le symbole S_i est donné par :

$$\hat{x}_{3|\tilde{x}_4=S_i} = \frac{y_3^{'}}{R_{33}} - S_i.\frac{R_{34}}{R_{33}}. \tag{2.60}$$

Par conséquent la probabilité $p(\tilde{x}_3=S_j|\tilde{x}_4=S_i)$ peut s'écrire comme suit :

$$p(\tilde{x}_3=S_j|\tilde{x}_4=S_i) = \int_{R_j} f_{\hat{x}_{3|\tilde{x}_4=S_i}}(x)dx. \tag{2.61}$$

où $f_{\hat{x}_{3|\tilde{x}_4=S_i}}$ représente la FDP du signal $\frac{y_3^{'}}{R_{33}}$ décalée par $-S_i.\frac{R_{34}}{R_{33}}$.

Dans les mêmes conditions d'une constellation BPSK avec des distributions gaussiennes, nous avons :

$$p(\tilde{x}_4=S_i) = \frac{1}{2}; \qquad S_i = \pm 1 \tag{2.62}$$

$$f_{\hat{x}_{3|\tilde{x}_4=S_i}}(x) = \frac{1}{2.\sqrt{2\pi}.\sigma_{ch}}\left(\exp(-\frac{(x-1+S_i.\frac{R_{34}}{R_{33}})^2}{2\sigma^2_{ch}}) + \exp(-\frac{(x+1+S_i.\frac{R_{34}}{R_{33}})^2}{2\sigma^2_{ch}})\right). \tag{2.63}$$

Il est alors possible, à partir des relations précédentes, d'exprimer la fonction de densité de probabilité du signal présent à l'entrée de l'opérateur de décision $\hat{x}_2$ de l'antenne n°2 en tenant

compte de tous les scénarios des sorties de décision des opérateurs des antennes n°3 et 4 situées en amont.

Afin de pouvoir déterminer les probabilités d'erreur de décision $P_{i,j}^2$ de l'antenne n°2, il est nécessaire d'estimer la FDP du bruit b_2 à l'entrée de cet opérateur de décision en tenant compte du bruit de quantification dû au format virgule fixe au niveau de l'antenne n°2 et de la propagation des erreurs de décision produites au niveau des antennes n°3 et 4. De façon similaire à ce qui a été exposé pour le cas de l'antenne n°3, nous modélisons la propagation de ce bruit au sein de la FDP du bruit b_2 en entrée de l'opérateur de décision situé sur la voie n°2. Nous considérerons également comme précédemment que le bruit b_2 possède plusieurs composantes, chacune caractérisant un scénario antérieur parfaitement déterminé au niveau des antennes n°3 et 4. Or, pour chaque antenne, nous avons trois scénarios possibles dans le cas d'une modulation de type BPSK (pas d'erreur, une erreur de type $P_{-1,+1}$ et une erreur de type $P_{+1,-1}$), ce qui conduit à un total de 9 composantes possibles. Par conséquent, la FDP du bruit b_2 à l'entrée de l'opérateur de décision situé au niveau de l'antenne n°2 est donnée par :

$$f_B^{(2)}(x) = \sum_{k=1}^{9} p(S_{BPSK}^k).f_{B|S_{BPSK}^k}^{(2)}(x). \tag{2.64}$$

Avec $p(S_{BPSK}^k)$ est la probabilité de l'occurrence du scénario S_{BPSK}^k. $f_{S_{BPSK}^k}$ est la FDP de la composante du bruit qui caractérise la propagation de l'erreur générée par ce scénario. A titre d'exemple, si l'on considère le scénario correspondant à une erreur de type $P_{-1,+1}$ et $P_{+1,-1}$ respectivement au niveau des antennes n°4 et 3, la probabilité de réalisation de ce scénario est alors donnée par :

$$P(\underbrace{(\tilde{x}_4^{vflo} = -1, \tilde{x}_4^{vfix} = 1), (\tilde{x}_3^{vflo} = -1, \tilde{x}_3^{vfix} = -1)}_{\substack{scenario \\ propagé}}) = P_{-1,1}^4.P_{1,-1}^3. \tag{2.65}$$

Par ailleurs, la FDP $f_{B|S_{BPSK}^k}^{(2)}(x)$ de la composante de bruit associée à ce scénario sera donnée par la FDP du bruit de quantification de moyenne décalée de $-(-1-1).\frac{R_{24}}{R_{22}} - (1-(-1)).\frac{R_{23}}{R_{22}}$.

En suivant cette stratégie, nous pouvons déterminer toutes les composantes qui modélisent la propagation des erreurs de décisions engendrées par l'arithmétique virgule fixe et par conséquent la FDP du bruit b_2 présent à l'entrée de l'opérateur de décision de l'antenne n°2 est déterminée.

A ce stade de notre analyse, nous avons tous les éléments nécessaires pour appliquer l'équation (2.17) afin d'exprimer les probabilités d'erreur de décision $P_{i,j}^2$. Afin d'évaluer numériquement les quantités impliquées dans cette approche mathématique, nous avons calculé les différentes expressions analytiques à l'aide de MATHEMATICA [50] qui est un logiciel de calcul formel. Les résultats ainsi obtenus sont donnés à la figure 2.20 où nous avons représenté la probabilité totale d'erreur de décision en fonction du rapport signal à bruit dans le cas d'une constellation de type BPSK lorsque les bruits de quantification et le bruit de canal sont considérés comme gaussiens (hypothèses similaires à celles faites lors de l'analyse de l'antenne n°3). D'après cette figure, nous remarquons la coïncidence entre la probabilité d'erreur de décisions obtenue à l'aide du modèle analytique et celle obtenue à partir de simulations de type Monte Carlo pour des RSB compris entre 0 et 10 dB. De même que dans le cas de l'antenne n°3, l'erreur maximale entre les deux probabilités estimées est de l'ordre de 2%.

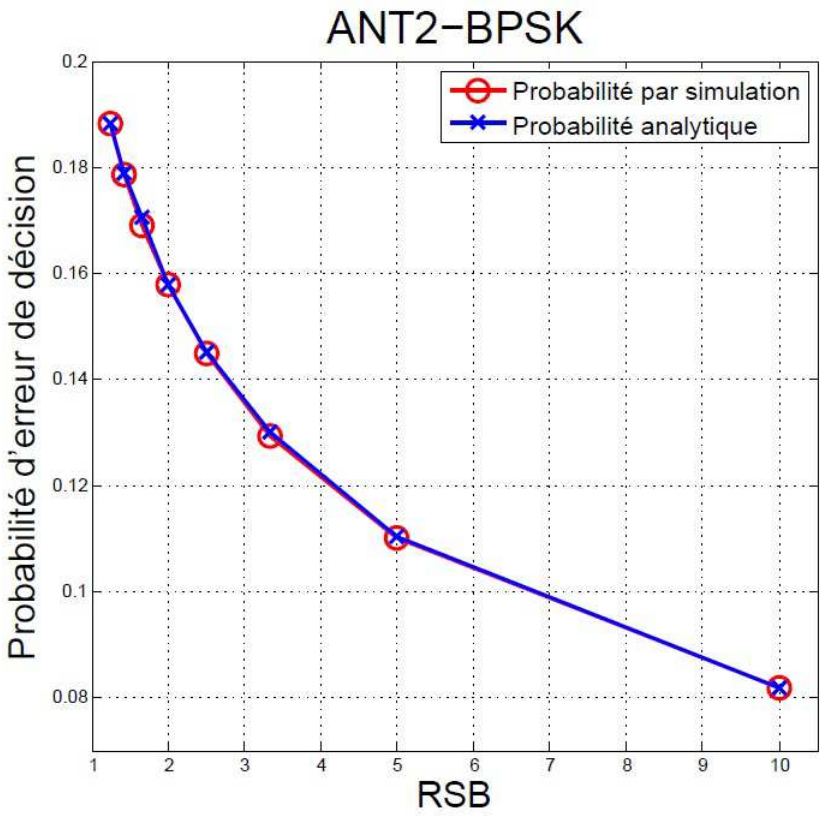

FIGURE 2.20 – Probabilité d'erreur de décision totale en fonction du rapport signal à bruit

Cas générique Dans le cas d'une constellation de type M-QAM, le signal à l'entrée de l'opérateur de décision de l'antenne n°2 est défini par la relation (2.56). En considérant de plus que les sorties des opérateurs de décision des antennes n°4 et n°3 soient respectivement les symboles S_i et S_j, nous obtenons :

$$\hat{x}_{2_{|\tilde{x}_4=S_i,\tilde{x}_3=S_j}} = \frac{y_2^{'}}{R_{22}} - S_i\frac{R_{24}}{R_{22}} - S_j\frac{R_{23}}{R_{22}}. \tag{2.66}$$

Par conséquent, la FDP du signal $\hat{x}_2$ à l'entrée de l'opérateur de décision de l'antenne n°2 est donnée par :

$$f_x^{(2)}(x) = \sum_{i=1}^{M}\sum_{j=1}^{M} p(\tilde{x}_4 = S_i, \tilde{x}_3 = S_j).f^{(2)}_{\hat{x}_{2_{|\tilde{x}_4=S_i,\tilde{x}_3=S_j}}}(x). \tag{2.67}$$

Avec $f^{(2)}_{\hat{x}_{2_{|\tilde{x}_4=S_i,\tilde{x}_3=S_j}}}(x)$ $(S_i = \pm 1; S_j = \pm 1)$ est la FDP du signal ($\frac{y_2^{'}}{R_{22}}$ reçu sur l'antenne n°2 décalé par $-S_i.\frac{R_{24}}{R_{22}} - S_j.\frac{R_{23}}{R_{22}}$. Les probabilités $p(\tilde{x}_4 = S_i, \tilde{x}_3 = S_j)$ sont déterminées de la même façon que le cas BPSK.

Il ne reste qu'à déterminer la FDP du bruit b_2 en entrée de l'opérateur de décision afin d'exprimer toutes les probabilités d'erreur de décision $P^2_{i,j}$ en tenant compte de la propagation des différents scénarios possibles d'erreurs de décision produites au niveau des antennes n°3 et n°4. Pour les antennes n°3 et n°4, il y a $M(M-1)$ erreurs de décision possibles causées par le format arithmétique virgule fixe pour chaque antenne excepté le scénario où il n'y a pas d'erreur de décision. Le nombre de scénarios possibles de propagation des erreurs des précédents blocs de décision jusqu'à l'opérateur de décision de l'antenne n°2 et $(M(M-1)+1)^2$. Nous avons donc :

$$f_B^{(2)}(x) = \sum_{k=1}^{(M(M-1)+1)^2} p(S_{M-QAM}^k).f_{B|S_{BPSK}^k}^{(2)}(x). \tag{2.68}$$

Les différentes probabilités de l'occurrence des scénarios $p(S_{M-QAM}^k)$ sont déterminées de la même façon que le cas BSPK.

Deuxième approche

L'approche précédente nécessite d'évaluer la FDP du signal et du bruit à l'entrée de chaque opérateur de décision pour chaque antenne de l'algorithme SSFE en tenant compte de tous les scénarios possibles de la propagation des erreurs de décision d'un opérateur à un autre dans la cascade considérée. La nature itérative d'une telle approche entraîne naturellement une complexité importante de l'implémentation associée dans le cas générique, et ceci d'autant plus que la constellation numérique utilisée possède un haut rendement spectral.

Afin de réduire cette complexité, nous proposons de modéliser la propagation des erreurs de décision dans la FDP du signal en entrée de l'opérateur de décision considéré dans la cascade en tenant compte de tous les scénarios possibles.

Antenne 3 Avec cette technique, nous prenons en compte la distance $d_{ij} = S_i - S_j$ lorsqu'une erreur de décision est produite au niveau de l'opérateur de décision de l'antenne n°4 avec une probabilité $P_{i,j}^4$. Soit $\hat{x}_{3,ij}$ le signal à l'entrée du bloc de décision de l'antenne n°3 qui prend en compte la différence d_{ij} entre les symboles impliqués dans l'erreur produite au niveau de l'antenne n°4 :

$$\hat{x}_{3,ij} = \frac{y_3^{'}}{R_{33}} - \frac{R_{34}}{R_{33}} d_{ij}. \tag{2.69}$$

La FDP $f_{\hat{x}_{3,st}}(x)$ du signal $\hat{x}_{3,ij}$ est égale à la FDP du $\frac{y_3^{'}}{R_{33}}$ décalée par $\frac{R_{34}}{R_{33}} d_{ij}$. Par conséquent la probabilité d'erreur de décision $P_{i,j}^3$ au niveau de l'opérateur de décision de l'antenne n°3 correspond à la somme pondérée des probabilités d'erreur de décision partielles $P_{i,j|s,t}^3$ sous contrainte de la propagation d'un scénario d'erreur de décision produit lors du bloc de décision de l'antenne n°4 correspondant à l'évènement aléatoire suivant : pour l'antenne n°4, décision d'un symbole S_t en arithmétique virgule fixe alors que ce symbole correspond à S_s en arithmétique flottante. La probabilité d'un tel évènement sera notée $P_{s,t}^4$ dans la suite du manuscrit. Chaque élément de cette somme est pondéré par la probabilité de l'occurrence du scénario qui lui correspond. Au niveau de l'antenne n°3, la probabilité d'erreur de décision $P_{i,j|s,t}^3$ est obtenue à partir de la relation (2.17) suivant :

$$P_{i,j|s,t}^{'3} = \int_{R_i} \int_{R_j} f_{\hat{x}_{3,st}}(x)) f_B^3(b-x) db dx. \tag{2.70}$$

Par conséquent les probabilités d'erreur de décision $P_{i,j}^3$ sont déterminées par :

$$P_{ij}^3 = \sum_{s,t=1}^{M} P_{s,t}^4 \int_{R_i} \int_{R_j} f_{\hat{x}_{3,st}}(x) f_B^3(b-x) db.dx \tag{2.71}$$

$$P_{ij}^3 = \int_{R_i} \int_{R_j} (\sum_{s,t=1}^{M} P_{s,t}^4 f_{\hat{x}_{3,st}}(x)) f_B^3(b-x) db dx. \tag{2.72}$$

La figure 2.21 représente, dans le cas d'une constellation de type 4-QAM, la probabilité d'erreur de décision au niveau de l'antenne n°3 estimée à partir de l'équation (2.72) mais également celle obtenue à partir de simulations de type Monte Carlo. La constellation 4-QAM est constituée

de 4 symboles modulés : $S_1 = 1 + i; S_2 = -1 + i; S_3 = -1 - i; S_4 = 1 - i$. Chaque symbole S_k est composé d'une composante en phase S_k^{re} et d'une composante en quadrature S_k^{im} qui prennent leurs valeurs de façon équiprobable dans l'alphabet binaire -1,+1. Nous considérons que les signaux reçus sur les antennes suivent une loi gaussienne dont la FDP est donnée par :

$$f_X(x,y) = \frac{1}{2.\pi.\sigma_{ch}^2}.\frac{1}{M}\sum_{i=1}^{M}\exp(\frac{-(x-S_i^{re})^2-(y-S_i^{im})^2}{2\sigma_{ch}^2}). \tag{2.73}$$

Nous considérons également que le bruit de quantification est un bruit blanc bidimensionnel dont la FDP est donnée par :

$$f_B(x,y) = \frac{1}{2.\pi.\sigma_b^2}.\exp(\frac{-(x)^2-(y)^2}{2\sigma_b^2}). \tag{2.74}$$

La figure 2.21 représente les probabilités d'erreur de décision P_{ij} évalués analytiquement et par simulation dans le cas d'une puissance du bruit de quantification de 0.5.

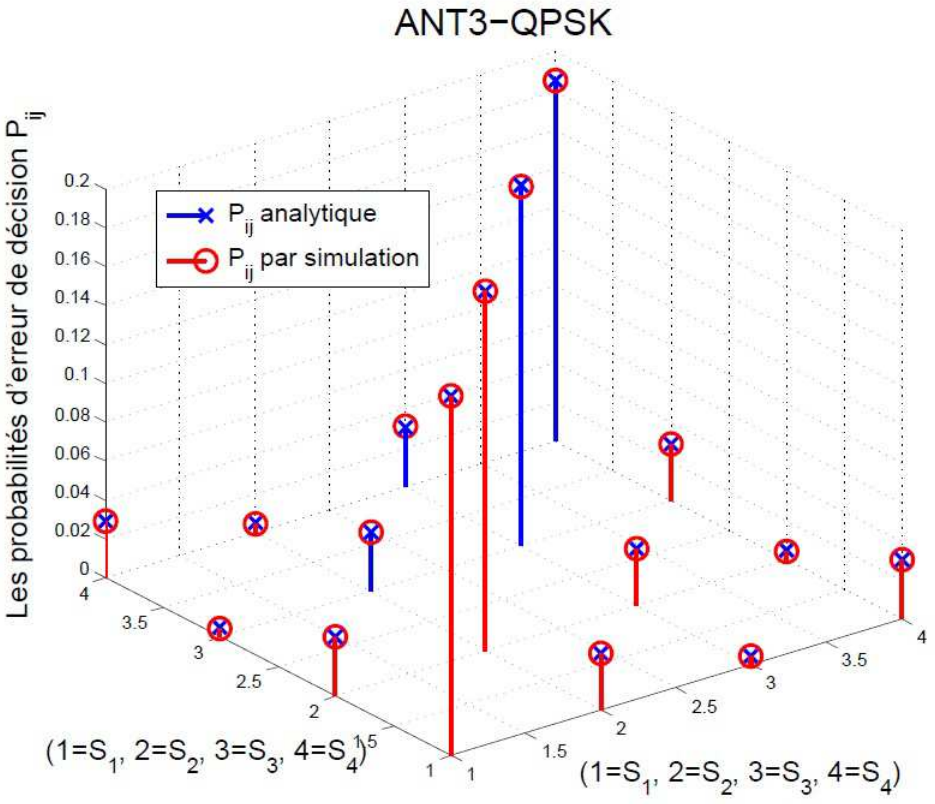

FIGURE 2.21 – Probabilité d'erreur de décision P_{ij}^3 dans le cas d'une puissance du bruit de quantification de 0.5

Nous pouvons remarquer que les P_{ii} sont les probabilités les plus élevées parce qu'elles sont en d'autres termes les probabilités de ne pas avoir une erreur de décision. Par contre les probabilités d'erreur de décision P_{S_1,S_3} et P_{S_3,S_1} sont quasiment égales et correspondent aux probabilités d'erreur de décision les plus faibles car les symboles S_1 et S_3 sont les symboles les plus éloignés de la constellation. C'est également le cas pour les probabilités P_{S_2,S_4} et P_{S_4,S_2}. De façon analogue aux résultats établis dans le cas de la première approche, nous représentons à la figure 2.22 la probabilité d'erreur de décision totale obtenue à partir de l'approche analytique présentée ci-avant et celle obtenue à partir de simulations de type Monte-Carlo.

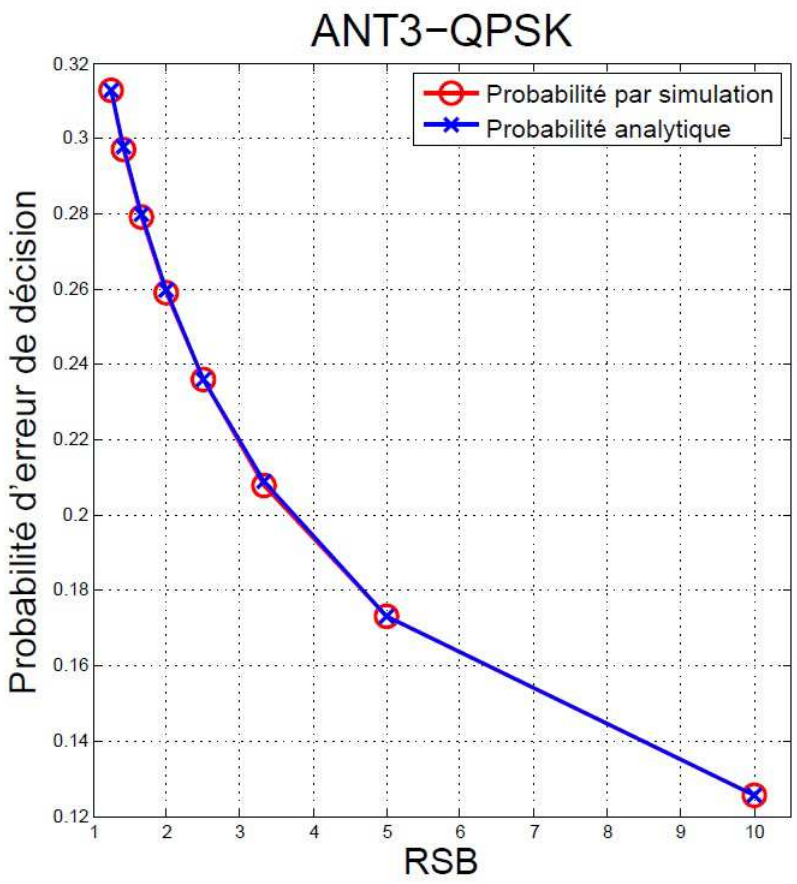

FIGURE 2.22 – Probabilité d'erreur au niveau de l'antenne n°3 de décision totale dans le cas d'une constellation 4-QAM

Antenne 2 En suivant le même raisonnement que précédemment, nous analysons ci-après les performances de l'algorithme SSFE au niveau de l'antenne de réception n°2. Soit $\hat{x}_{2,ij,st}$ le signal présent à l'entrée de l'opérateur de décision de l'antenne n°2 sous contrainte d'un scénario comportant une erreur de décision provenant des opérateurs de décision situés en amont (antenne n°3 et/ou n°3). En appelant $P_{s,t}^{k}$ la probabilité de commettre une erreur de décision correspondant à l'évènement aléatoire suivant : pour l'antenne n°k, décision d'un symbole S_t en arithmétique virgule fixe alors que ce symbole correspond à S_s en arithmétique flottante, nous avons :

$$\hat{x}_{2,ij,st} = \frac{y_2^{'}}{R_{22}} - \frac{R_{24}}{R_{22}} d_{ij} - \frac{R_{23}}{R_{22}} d_{st}. \tag{2.75}$$

La FDP de $\hat{x}_{2,ij,st}$ est égale à la FDP de $\frac{y_2^{'}}{R_{22}}$ décalée par $-\frac{R_{24}}{R_{22}} d_{ij} - \frac{R_{23}}{R_{22}} d_{st}$. Par conséquent les probabilités d'erreur de décision P_{ij}^{2} sont égales à la somme pondérée des probabilités partielles $P_{i,j|st,pq}^{'2}$ qui sont déterminées en appliquant le modèle de l'équation (2.17) à la FDP $f_{\hat{x}_{2,pq,st}}$ de $\hat{x}_{2,pq,st}$ et la FDP du bruit de quantification en les multipliant par la probabilité de l'occurrence de ce scénario $P_{p,q}^{4} P_{s,t}^{3}$, soit :

$$P_{ij}^{2} = \int_{R_i} \int_{R_j} \Big(\sum_{p,q,s,t=1}^{M} P_{p,q}^{4} P_{s,t}^{3} f_{\hat{x}_{(2),pq,st}}(x) \Big) f_B^{(2)}(b-x) db dx. \tag{2.76}$$

Antenne n°1 Pour l'antenne n°1 qui est la dernière antenne du système MIMO considéré, nous utilisons la même technique pour déterminer les probabilités d'erreur de décision P_{ij}^{1}. Par conséquent nous définissons $f_1(x)$ la FDP du signal à l'entrée de l'opérateur de décision de

l'antenne n°1, en tenant compte de tous les scénarios d'erreurs et de bonnes décisions possibles au niveau des autres antennes précédentes n°2, 3 et 4, par :

$$f^1(x) = \sum_{p,q,s,t,v,w=1}^{M} P^4_{p,q} P^3_{s,t}.P^2_{v,w} f_{\hat{x}_{1,pq,st,vw}}(x). \tag{2.77}$$

où $f_{\hat{x}_{1,pq,st,vw}}(x)$ désigne la FDP du signal $\hat{x}_{1,pq,st,vw}$ à l'entrée de l'opérateur de décision de l'antenne n°1 sous contrainte d'un scénario d'erreur de décision au niveau des antennes situées en amont dont la probabilité d'occurrence est $P^4_{p,q}.P^3_{s,t}.P^2_{v,w}$. Elle est égale à la FDP du signal reçu $\frac{y'_1}{R_{11}}$ décalée par $-\frac{R_{14}}{R_{11}}.d_{pq} - \frac{R_{13}}{R_{11}}.d_{st} - \frac{R_{12}}{R_{11}}.d_{vw}$. Par conséquent les probabilités d'erreur de décision $P^1_{i,j}$ sont définies par :

$$P^1_{ij} = \int_{R_i}\int_{R_j} f_1(x) f^1_B(b-x) db dx. \tag{2.78}$$

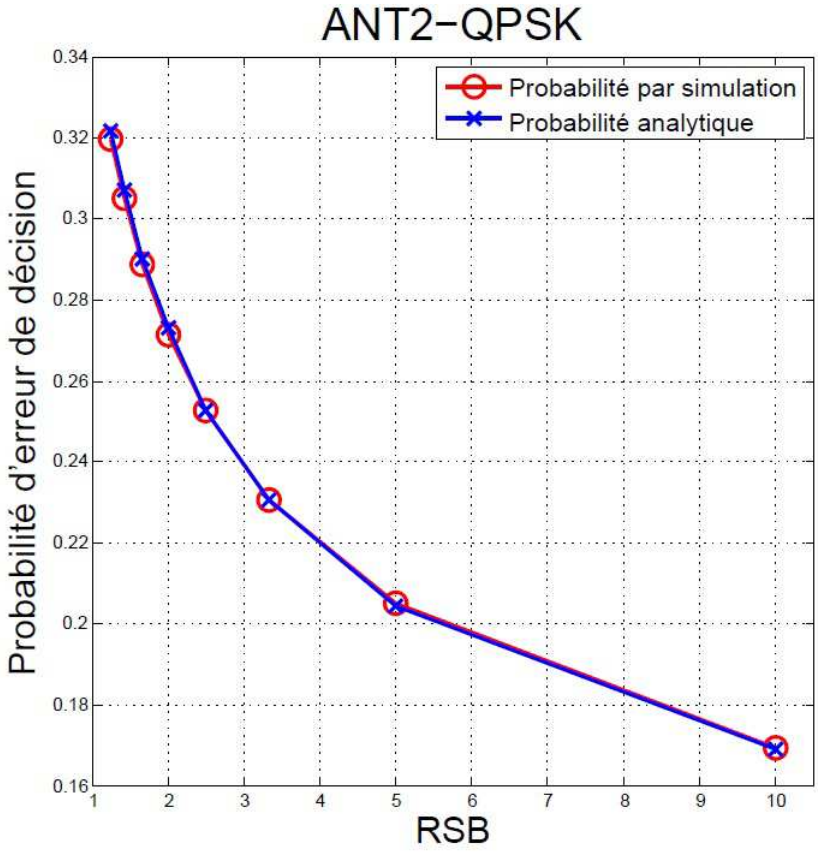

FIGURE 2.23 – Probabilité d'erreur de décision totale dans le cas d'une constellation 4-QAM

Comme dans le cas de l'antenne n°3, nous représentons sur les figures 2.23 et 2.25 les probabilités d'erreur de décision au niveau des antennes n°2 et n°1. Les conditions d'expérimentations sont identiques à celles sélectionnées précédemment pour l'étude de l'antenne n°3. Ainsi, les signaux reçus correspondent aux symboles 4-QAM perturbés un bruit blanc additif, bi-variant, gaussien, et centré. De même, le bruit de quantification est modélisé dans ces expériences par un bruit blanc gaussien bidimensionnel de moyenne nulle. La matrice R utilisée dans ces expériences est donnée par :

$$R = \begin{pmatrix} -0.4993 & -0.1413 - 0.1131i & 0.3099 - 0.3319i & -0.4374 + 0.3375i \\ 0 & 0.48 & -0.025 - 0.1395i & 0.1814 + 0.1514i \\ 0 & 0 & 0.403 & -0.0303 - 0.1145i \\ 0 & 0 & 0 & -0.4766 \end{pmatrix} \tag{2.79}$$

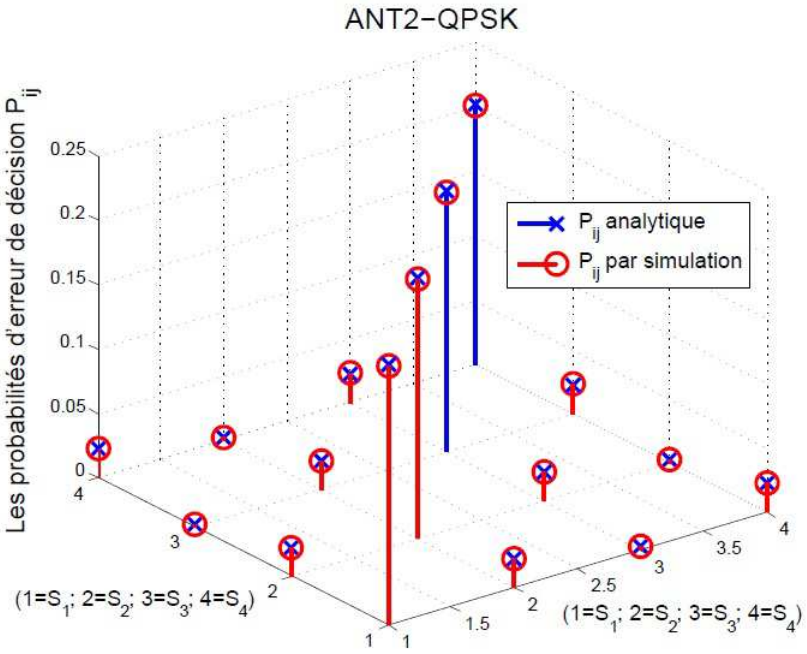

FIGURE 2.24 – Probabilité d'erreur de décision P_{ij}^2 dans le cas d'une constellation 4-QAM

Nous remarquons que la probabilité d'erreur de décision totale causée par l'arithmétique virgule fixe augmente d'une antenne à l'autre de la cascade d'opérateurs de décision pour un rapport signal à bruit donné en entrée du récepteur. Ce résultat était prévisible puisque la propagation des erreurs de décision non lisses augmente la probabilité d'avoir d'autres erreurs de décision au sein des opérateurs de décision correspondant aux antennes situées en aval de l'antenne courante. En effet, le bruit à l'entrée de l'opérateur de décision considéré n'est pas seulement constitué du bruit de quantification du format virgule fixe, mais aussi d'autres bruits modélisant les erreurs engendrées par les opérateurs de décision situés en amont de l'antenne courante.

Nous pouvons également vérifier la robustesse de notre modèle analytique en comparant les probabilités d'erreur de décision $P_{i,j}$ analytiques et celles obtenues par simulation. La somme de toutes les probabilités $P_{i,j}$ est égale à 1, ce qui valide le modèle analytique proposé. De plus d'après les figures, il est clair que les probabilités obtenues analytiquement sont très proches des probabilités par simulation avec une marge d'erreur très faible (inférieure à 1%).

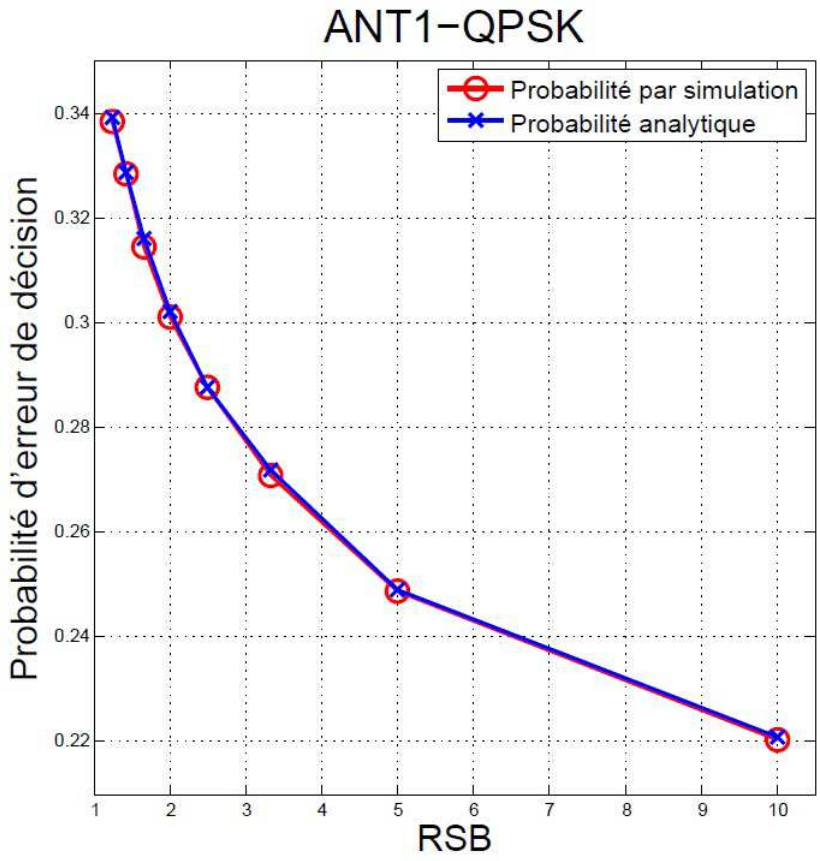

FIGURE 2.25 – Probabilité d'erreur de décision totale dans le cas d'une constellation 4-QAM

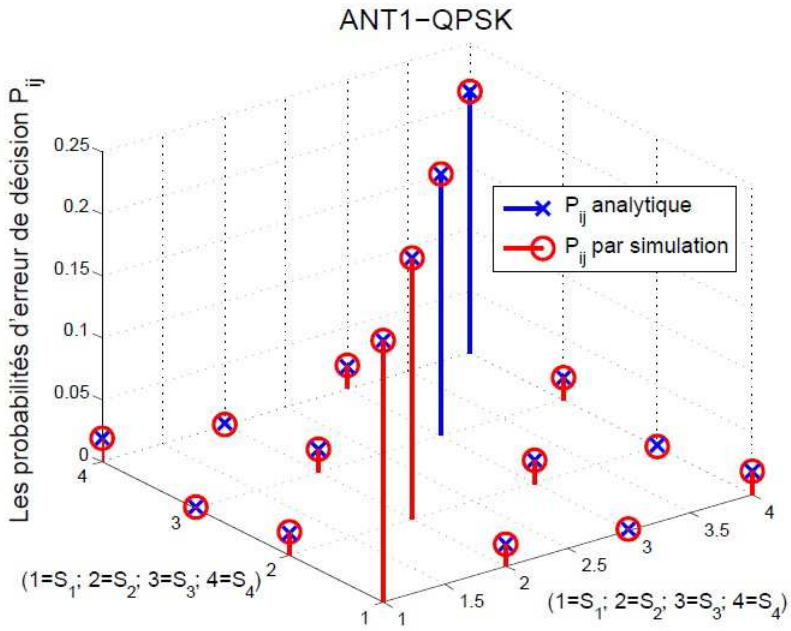

FIGURE 2.26 – Probabilité d'erreur de décision P_{ij}^{1} dans le cas d'une constellation 4-QAM

Afin de montrer l'intérêt du modèle analytique proposé, nous citons un ordre de grandeur du temps d'exécution mis pour estimer l'ensemble des probabilités d'erreur de décision $P_{i,j}^{1}$ au niveau de l'opérateur de décision de l'antenne n°1. Le temps d'exécution nécessaire pour évaluer

numériquement les probabilités $P_{i,j}^{1}$ dans le cas d'une constellation 4-QAM est de 6 minutes et 37 secondes. Par contre, le temps nécessaire à la simulation de toute la chaîne de transmission MIMO avec un détecteur SSFE en arithmétique virgule flottante et en arithmétique virgule fixe est de l'ordre 47 minutes et 56 secondes avec les mêmes ressources informatiques et les mêmes capacités de calcul et mémoire (Processeur Intel i7 2.50 GHz, 4 coeurs, RAM : 4 Go).

2.5 Conclusion

Dans ce chapitre, nous avons caractérisé les opérateurs *non lisses* et leurs identifications. Dans un premier temps, nous avons présenté un modèle analytique pour l'évaluation de la précision d'un opérateur de décision *non lisse* en estimant la probabilité d'erreur de décision causée par la quantification du format virgule fixe. Ensuite, nous avons analysé la propagation des erreurs de décision au sein d'un système contenant plusieurs opérateurs *non lisses* de décision afin d'être en mesure de proposer un modèle général pour évaluer la précision des systèmes contenant une cascade d'opérateurs de décision en tenant compte de la propagation des erreurs et de tous les scénarios d'erreurs possibles dans la cascade. En outre, nous avons développé ce modèle dans le cas de l'algorithme SSFE en proposant deux approches pour estimer les probabilités d'erreur de décision au sein de chaque opérateur de décision impliqué dans la cascade au niveau de l'algorithme.

Nous avons réussi à proposer un modèle analytique pour évaluer la cascade d'opérateurs de décision et résoudre le problème d'évaluation de l'influence de la représentation en virgule fixe pour une certaine classe d'algorithmes de communications numériques. Cependant, d'autres algorithmes, également constitués par des blocs d'opérations lisses et des opérateurs de décision, ne peuvent être analysés à l'aide de ce modèle en raison de leur structure récursive. Le récepteur à égaliseur à retour de décision (DFE pour *Decision Feedback Equalizer*) en est un exemple important. Dans le prochain chapitre nous proposerons des modèles analytiques pour évaluer ce genre de configuration et nous montrerons comment les modèles proposés peuvent contribuer à une implémentation matérielle efficace en termes de consommation de ressources matérielles et d'énergie.

Chapitre 3

Évaluation de la précision de l'itération d'opérateur de décision

Sommaire

Nous avons présenté dans le chapitre précédent des modèles analytiques pour évaluer la précision des opérateurs non lisses de décision et leurs cascades en estimant la probabilité d'erreur de décision causée par le bruit de quantification de l'arithmétique virgule fixe et également la propagation des erreurs de décision d'un opérateur de décision à un autre dans un système constitué d'une cascade de ces opérateurs, tel par exemple qu'un récepteur utilisant un décodage sphérique de type SSFE. Néanmoins, d'autres systèmes de communication numérique et de traitement de signal, tels que les algorithmes d'égalisation à retour de décision ou bien encore le turbo décodage, présentent des itérations d'opérateurs non lisses au sein d'une structure récursive ce qui est encore équivalent à considérer une cascade infinie d'opérateurs de décision. Par conséquent, il est intéressant de se demander dans quelle mesure les modèles proposés au chapitre précédent pour l'évaluation de la précision des applications contenant des cascades d'opérateurs de décision sont applicables pour évaluer les itérations d'opérateurs de précision lisses et sous quelles contraintes ?

Donc, il est nécessaire d'analyser la propagation des erreurs de décision non lisses dans ces applications en modélisant cette propagation du bruit et des erreurs de décision. En effet, il est fondamental de caractériser l'effet des erreurs de décision dans des applications d'itérations d'opérateurs non lisses de décision spécifiquement au moment où l'application converge vers les performances visées par le concepteur. A notre connaissance, il n'existe aucun modèle analytique disponible dans la littérature permettant d'évaluer les performances de ces applications en virgule fixe pour un format bien déterminé. Par conséquent, il est important de proposer des modèles analytiques permettant de caractériser les erreurs de décision lors de la phase asymptotique

de ces applications comportant des itérations d'opérateurs non lisses. En effet, il est nécessaire de pouvoir disposer de tels outils afin de sélectionner le format optimal de représentation en arithmétique virgule fixe en termes de probabilité d'erreur de décision minimale mais aussi en termes d'optimisation de l'implémentation matérielle à travers la minimisation des ressources et de la consommation en énergie.

Ce chapitre vise à présenter des modèles analytiques permettant d'évaluer la précision des applications formées de structures récursives d'opérateurs non lisses. Nous présenterons dans un premier temps la problématique de la propagation des erreurs dans le cas de ces applications. Ensuite deux modèles analytiques pour évaluer la précision des itérations d'opérateurs de décision seront représentés par deux approches différentes. Nous validerons ces modèles dans la troisième section en les appliquant dans le cas de l'égaliseur à retour de décision DFE (*Decision Feedback Equaliser*). Dans la dernière section de ce chapitre, nous illustrons l'intérêt de ces modèles analytiques dans le cas particulier de l'optimisation du format en virgule fixe du DFE pour une implémentation sur FPGA.

3.1 Problématique

Nous considérons un système représenté par la figure 3.1 constitué par des blocs d'opérations lisses et une itération d'opérateurs de décision. Ce système est constitué d'une opération de décision dont la sortie influence la prochaine décision jusqu'à une convergence de l'application du système.

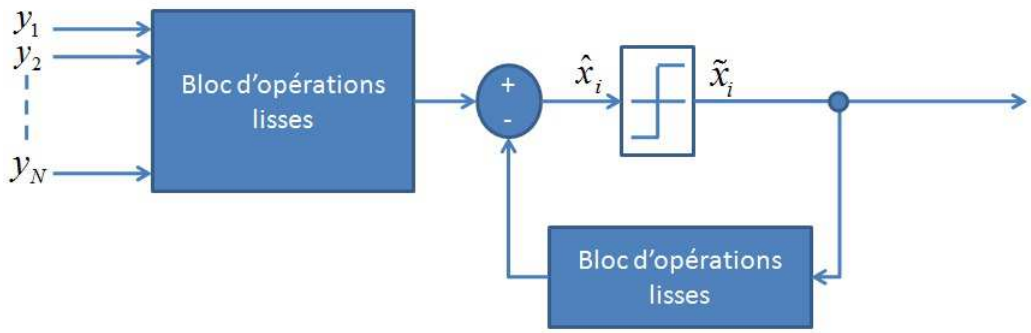

FIGURE 3.1 – Système contenant une itération d'opérateur de décision

Ce système est composé de N entrées $Y = [y_1, y_2, ..., y_N]$ qui sont perturbées par les bruits de quantification $B = [b_1, b_2, ..., b_N]$. Chaque bruit b_i perturbe l'entrée y_i.

Soient $\hat{x}_i$ et $\tilde{x}_i$ respectivement l'entrée et la sortie de l'opérateur de décision à la i^{eme} itération.

Le second bloc d'opérations lisses dont la fonction est $L_2(\hat{x}_{i-1}, \hat{x}_{i-2}, ..., \hat{x}_{i-T_l})$ possède à son entrée les sorties de l'opérateur de décision aux itérations précédant l'itération actuelle avec une mémoire de taille T_l éléments.

Dans la suite du chapitre, nous nommerons $\hat{x}_i^{vflo}$ et $\hat{x}_i^{vfix}$ les entrées de l'opérateur de décision à l'itération n°i respectivement en arithmétique virgule flottante et virgule fixe. De même, $\tilde{x}_i^{vflo}$ et $\tilde{x}_i^{vfix}$ correspondent aux sorties de l'opérateur de décision à l'itération n°i respectivement en arithmétique virgule flottante et virgule fixe.

3.1.1 Caractéristiques de la propagation du bruit

Dans le chapitre précédent, nous avons présenté un modèle analytique pour l'évaluation de la précision d'un opérateur de décision non lisse. Ce modèle n'est plus valide pour évaluer la précision d'un système constitué d'une structure récursive de plusieurs opérateurs de décision. Par conséquent, il est nécessaire de modifier ce modèle et de l'adapter aux contraintes de ces

systèmes en tenant compte de tous les scénarios d'erreurs possibles et leurs propagations dans le système considéré. De même, il est fondamental de caractériser la propagation des erreurs de décision et leur impact non seulement sur la décision de l'itération en cours, mais aussi lorsque le système converge. Pour cette raison, il est important de modéliser cette propagation dans le mode de convergence des systèmes comportant des itérations d'opérateurs non lisses. A l'itération courante (n°i), le signal présent en entrée de l'opérateur de décision est donné par :

$$\begin{aligned}\hat{x}_i^{vfix} &= L_1(\tilde{x}_{i-1}^{vfix}, \tilde{x}_{i-2}^{vfix}, ..., \tilde{x}_{i-T_l}^{vfix}) + L_2(y_1^{vfix}, ..., y_N^{vfix}) \\ &= L_1(\tilde{x}_{i-1}^{vflo}, \tilde{x}_{i-2}^{vflo}, ..., \tilde{x}_{i-T_l}^{vflo}) + L_2(y_1^{vflo}, ..., y_N^{vflo}) \\ &+ l(b_{i-T_l}, ..., b_{i-1}, \tilde{x}_{i-1}^{vfix}, \tilde{x}_{i-2}^{vfix}, ..., \tilde{x}_{i-T_l}^{vfix}, \tilde{x}_{i-1}^{vflo}, \tilde{x}_{i-2}^{vflo}, ..., \tilde{x}_{i-T_l}^{vflo}) + b_L \\ &= \hat{x}_i^{vflo} + L_{pro}(B^i, \tilde{X}_{vflo}^i, \tilde{X}_{vfix}^i) + b_L \end{aligned} \quad (3.1)$$

où B^i est le vecteur des perturbations à l'itération i. L_{pro} représente la fonction de propagation du bruit de quantification lors des itérations précédentes en respectant la taille T_l de la mémoire du bloc lisse situé après l'opérateur de décision de l'itération. Le signal b_L correspond au bruit ajouté par les opérateurs lisses de l'implémentation en arithmétique virgule fixe des fonctions lisses L_1 et L_2. Enfin, $\tilde{X}_{vfix}^i = [\tilde{x}_{i-1}^{vfix}, \tilde{x}_{i-2}^{vfix}, ..., \tilde{x}_{i-T_l}^{vfix}]$ et $\tilde{X}_{vflo}^i = [\tilde{x}_{i-1}^{vflo}, \tilde{x}_{i-2}^{vflo}, ..., \tilde{x}_{i-T_l}^{vflo}]$ désignent les vecteurs contenant les sorties des opérateurs de décision enregistrées dans la mémoire du bloc lisse respectivement en arithmétique virgule fixe et virgule flottante. La propagation des erreurs à la sortie de l'opérateur de décision des itérations précédentes possède des caractéristiques non lisses. Par conséquent, la fonction L_{pro} de la propagation des bruits de quantification doit être évaluée pour chaque erreur de la première itération jusqu'à l'itération courante en cours d'évaluation. Cette évaluation est nécessaire jusqu'à la convergence du système où l'impact de la quantification doit être négligeable si l'on souhaite obtenir une évaluation précise des paramètres de ce système.

3.2 Modèle analytique proposé

L'objectif de la caractérisation de la propagation des erreurs de quantification d'une itération à l'autre consiste à évaluer les différentes probabilités d'erreur de décision P_{ij} lorsque le système converge. En effet, la convergence du système conduit à une convergence au niveau des probabilités d'erreur de décision d'une itération à une autre. En d'autres termes, l'évolution de la probabilité d'erreur de décision d'une itération à une autre est négligeable lors de la phase de convergence asymptotique.

Deux approches différentes sont proposées pour résoudre cette problématique. La première approche est basée sur une résolution d'un système non linéaire en utilisant la méthode de Newton-Raphson qui sera détaillée dans la section suivante. La deuxième approche est fondée sur le principe de détermination d'une borne supérieure de l'erreur de décision dans la zone de convergence du système.

3.2.1 Approche basée sur la résolution d'un système non linéaire à l'aide de l'algorithme de Newton-Raphson

Considérons v_{vflo} et v_{vfix} deux instances des valeurs prises respectivement par $\tilde{X}_{vflo}^i$ et $\tilde{X}_{vfix}^i$. Par conséquent, la fonction de propagation s'écrit sous sa forme particulière l_{pro}^v qui est réduite à une fonction dépendante du vecteur des perturbations B_i. Le signal en virgule fixe à l'entrée de l'opérateur de décision au cours de l'itération i s'écrit par la relation suivante :

$$\begin{aligned}\hat{x}_i^{vfix} &= \hat{x}_i^{vflo} + l_{pro}^v(B_i) + b_L \\ &= \hat{x}_i^{vflo} + b_t. \end{aligned} \quad (3.2)$$

Les vecteurs des données en entrée de l'opérateur de décision en arithmétique virgule fixe et virgule flottante représentent conjointement un scénario S déterminé par les conditions en

entrée et la propagation des signaux à travers les blocs lisses L_1 et L_2. Les bruits des blocs lisses engendrés par l'implémentation en arithmétique virgule fixe sont intégrés au sein du paramètre b_t. En ce qui concerne l'opérateur de décision, nous considérons dans la suite que, à chaque itération, celui-ci effectue une décision dans un alphabet correspondant aux M symboles constituant la constellation de la modulation utilisée. Ainsi, le vecteur $\tilde{X}^i$ peut prendre $C = M^{T_l}$ combinaisons possibles. Le vecteur V^i défini par $V^i = [v_{vflo}, v_{vfix}]$ représente un scénario bien déterminé des valeurs particulières prises par les vecteurs $\tilde{X}^i_{vflo}$ et $\tilde{X}^i_{vfix}$. Il existe $C \times C$ scénarios possibles. Par conséquent, après N itérations, la probabilité $P^{S^k}_{i,j}$ d'erreur de décision est déterminée comme dans le cas de la cascade d'opérateurs de décision par l'équation suivante :

$$P^{S^k}_{i,j} = \int_{R_i}\int_{R_j} f^{S^k}_x(x) f_b(b-x) dbdx. \tag{3.3}$$

Cette probabilité représente la probabilité que la valeur de $\tilde{x}_N$ corresponde à S_i en arithmétique virgule flottante et à S_j en arithmétique virgule fixe sous la la condition de la réalisation d'un scénario S^k. La fonction $f^{S^k}_x(x)$ représente la FDP du signal $\hat{x}^{vflo}_N$ correspondant à la réalisation du scénario S^k. De façon analogue, $f_b(b)$ correspond à la FDP du bruit total b_t. La probabilité Pi, j d'erreur de décision $P_{i,j}$ à la sortie de l'opérateur de décision au cours de l'itération N est déterminée en considérant tous les scénarios possibles de propagation des erreurs de quantification. Par conséquent, la probabilité d'erreur de décision $P^N_{i,j}$ à la sortie de l'opérateur de décision de l'itération N est une somme des probabilités $P^{S^k}_{i,j}$ d'erreurs de décision sous la contrainte du scénario S^k pondérées par la probabilité P_{S^k} de réalisation de ce scénario.

$$\begin{aligned} P^N_{i,j} &= \sum_{k=1}^{C^2} P_{S^k} P^{S^k}_{i,j} \\ &= \sum_{k=1}^{C^2} \int_{R_i}\int_{R_j} P_{S^k} f^{S^k}_x(x) f_b(b-x_)dbdx. \end{aligned} \tag{3.4}$$

La détermination des probabilités P_{s^k} est faite de la même façon que dans le cas d'une cascade d'opérateurs de décision en respectant la taille de la mémoire T_l du bloc lisse sauvegardant les sorties de l'opérateur de décision obtenues lors des itérations précédentes. Par conséquent, la probabilité P_{s^k} de l'occurrence du scénario k est donnée par :

$$P_{S^k} = P((\tilde{x}^{vflo}_{N-1} = S_{i_{N-1}}, \tilde{x}^{vfix}_{N-1} = S_{j_{N-1}}) \quad et \quad ... \quad et \quad (\tilde{x}^{vflo}_{N-T_l} = S_{i_{N-T_l}}, \tilde{x}^{vfix}_{N-T_l} = S_{j_{N-T_l}})). \tag{3.5}$$

Les paramètres S_{i_r} et S_{j_r} représentent les valeurs prises par les sorties de l'opérateur de décision respectivement en arithmétique virgule flottante et virgule fixe au cours de l'itération r. Considérons le cas où les sorties de l'opérateur de décision ne sont pas corrélées. Par conséquent la probabilité de l'occurrence du scénario k est le produit des probabilités d'erreur de décision $P^k_{i_r,j_r} = P(\tilde{x}^{vflo}_r = S_{i_r}, \tilde{x}^{vfix}_r = S_{j_r})$:

$$P_{S^k} = \prod_{r=N-T_l}^{N-1} P^k_{i_r,j_r}. \tag{3.6}$$

Ainsi, la probabilité d'erreur de décision $P^N_{i,j}$ s'écrit sous la forme suivante :

$$P^N_{i,j} = \sum_{k=1}^{C^2} \prod_{r=N-T_l}^{N-1} P^k_{i_r,j_r}.P^{S^k}_{i,j} = P^{N-1}_{i,j} + \beta \tag{3.7}$$

où β désigne une quantité positive. Par conséquent, la probabilité d'erreur de décision $P_{i,j}$ converge puisqu'elle constitue une suite croissante et majorée. L'objectif est d'évaluer cette probabilité analytiquement dans le mode de convergence. Nous pouvons utiliser l'approche analytique

de la cascade d'opérateurs de décision pour déterminer analytiquement la probabilité d'erreur de décision dans la zone de convergence. Par contre, lorsqu'il s'agit d'un nombre assez élevé d'itérations pour atteindre le mode de convergence, cette approche nécessite de très grandes capacités de calculs pour prendre en compte tous les scénarios possibles. Par conséquent, il est nécessaire de trouver une autre piste afin d'évaluer analytiquement cette probabilité d'erreur de décision lorsque le système converge. Soit P^k le vecteur des probabilités d'erreur de décision $P^k_{i,j}$ à l'itération k défini par :

$$P^k = [P^k_{1,1}, P^k_{1,2}, ..., P^k_{i,j}, ..., P^k_{M,M}]^T. \tag{3.8}$$

Lors de la phase de convergence asymptotique, on a $P^k = P$ quel que soit $k > k_0$, avec P représentant le vecteur des probabilités d'erreur de décision. Par conséquent, l'équation (3.7) conduit à un système non linéaire de M^2 inconnues sous la conditions que $\sum_{i,j=1}^{M} P_{i,j} = 1$. En conséquence, le problème d'évaluation de la précision conduit à la résolution d'un système non linéaire à M^2 inconnues défini par :

$$P = R_P.P \tag{3.9}$$

avec R_P représentant une matrice de dimension $M^2 \times M^2$ et qui dépend du vecteur P ce qui rend le système (3.9) non linéaire.

Afin de résoudre numériquement ce problème non linéaire, nous avons sélectionné l'algorithme de Newton-Raphson en raison de la simplicité de sa mise en œuvre, mais aussi pour son efficacité et sa rapidité. Considérons un système non linéaire de dimension n défini par :

$$f_i(x_1, x_2, ..., x_n) = 0; 1 \leq i \leq n. \tag{3.10}$$

Soit $X = (x_1, x_2, ..., x_n)$. Au voisinage de X, nous pouvons écrire un développement de Taylor à l'ordre 1 pour chaque fonction f_i :

$$f_i(X + \delta X) = f_i(X) + \sum_{j=1}^{n} \frac{df_i}{dx_j}(X)\delta x_j + O(\|\delta X\|^2). \tag{3.11}$$

Le principe de l'algorithme de Newton-Raphson consiste à considérer que l'on puisse connaître un vecteur X qui se situe dans le le voisinage de la solutions optimale. Ensuite, nous cherchons δX tel que $X + \delta X$ s'approche de la solution en négligeant les termes de second ordre dans le développement de Taylor. Nous itérons ce processus, i.e. $X < -X + \delta X$, jusqu'à ce que le terme correctif δX devienne négligeable ce qui conduit à la phase de convergence asymptotique. Par conséquent, le système non linéaire est réduit à un système linéaire avec n equations et n inconnues qui sont les éléments du vecteur δX. Le détail de l'algorithme Newton-Raphson est explicité dans l'annexe. Afin de valider cette approche, nous considérons le système défini par la figure 3.2.

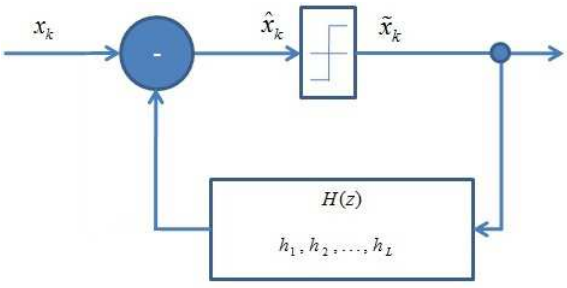

FIGURE 3.2 – Système composé d'itération d'opérateurs de décision

Ce système est constitué d'un opérateur de décision en itération avec un filtre FIR de L coefficients. Le signal $\hat{x}_k$ à l'entrée de l'opérateur de décision au cours de l'itération k dépend

non seulement du signal reçu x_k mais aussi des sorties de l'opérateur de décision $\tilde{x}$ au cours des itérations précédentes sous contrainte de la taille de la mémoire du filtre L. Le signal $\hat{x}_k$ est défini par :

$$\hat{x}_k = x_k - \sum_{l=1}^{L} h_l.\tilde{x}_{k-l} \tag{3.12}$$

où h_i désignent les coefficients du filtre H. Nous visons à déterminer la probabilité d'erreur de décision dans le mode de convergence du système en s'appuyant sur l'approche explicitée ci dessus. Soit z_k^e le signal qui prend en compte le signal x_k et soit $\Delta_{i,j}$ la différence entre les symboles impliqués dans l'erreur produite dans les itérations précédentes. Le signal z_k^e s'écrit par :

$$z_k^e = x_k - \sum_{l=1}^{L} h_l.\Delta_{i_l,j_l}. \tag{3.13}$$

Par conséquent, la FDP $f_{z_k^e}(x)$ du signal z_k^e dépend des probabilités d'erreur de décision précédentes P_{i_1,j_1}^{k-1},P_{i_2,j_2}^{k-2},...,P_{i_k,j_k}^{0}. Pour $k \leq L$, Cette FDP s'écrit par :

$$f_{z_k^e}(x) = \sum_{i_1=1}^{M}\sum_{j_1=1}^{M} ... \sum_{i_L=1}^{M}\sum_{j_L=1}^{M} P_{i_1,j_1}^{k-1}...P_{i_L,j_L}^{k-L}.f_{z_{i_1,j_1,...i_L,j_L}^{k}}(x) \tag{3.14}$$

avec $f_{z_{i_1,j_1,...i_k,j_k}^{k}}(x)$ est la FDP du signal x_k dont la moyenne est décalée par $\sum_{l=1}^{k} h_l.\Delta_{i_l,j_l}$. La probabilité d'erreur de décision $P_{i,j}^k$ pour l'itération k est donnée par :

$$P_{i,j}^k = \int_{R_i}\int_{R_j} f_{z_k^e}(x).f_B(b-x)db.dx. \tag{3.15}$$

Dans le mode de convergence où $P^k = P$ quelque soit $k > k_0$, nous définissons le signal z^e par le signal qui prend en compte la propagation des erreurs de décision dont la FDP est défini par :

$$f_{z^e}(x) = \sum_{i_1=1}^{M}\sum_{j_1=1}^{M} ... \sum_{i_L=1}^{M}\sum_{j_L=1}^{M} P_{i_1,j_1}...P_{i_L,j_L}.f_{z_{i_1,j_1,...i_L,j_L}}(x). \tag{3.16}$$

La probabilité d'erreur de décision dans le mode de convergence s'écrit alors suivant :

$$P_{i,j} = \int_{R_i}\int_{R_j} \sum_{i_1=1}^{M}\sum_{j_1=1}^{M} ... \sum_{i_L=1}^{M}\sum_{j_L=1}^{M} P_{i_1,j_1}...P_{i_L,j_L}.f_{z_{i_1,j_1,...i_L,j_L}}(x).f_B(b-x)db.dx. \tag{3.17}$$

Nous obtenons ainsi un système de la forme $P = R_P.P$ dont la résolution peut être obtenue par la méthode de Newton-Raphson. Pour tester ce modèle générique, nous utilisons des signaux reçus x_k composés de M composantes gaussiennes correspondant aux M symboles de la constellation M-QAM perturbés par la présence d'un bruit de canal blanc, bi-variant gaussien de variance σ_b^2 et centré.

$$f_{x_k}(x) = \sum_{j=1}^{M} \frac{1}{M}\frac{1}{\sqrt{2\pi}\sigma_y}\exp(-\frac{(x-S_j)^2}{2\sigma_y^2}). \tag{3.18}$$

$$f_B(x) = \frac{1}{\sqrt{2\pi}\sigma_b}\exp(\frac{-(x)^2}{2\sigma_b^2}). \tag{3.19}$$

Afin d'évaluer l'influence de la variance du bruit de quantification sur le système, nous modifions le nombre de bits du format de quantification. Ainsi, pour chaque puissance de bruit, l'exécution de

l'algorithme du système permet d'obtenir les signaux en sortie du système en arithmétique virgule fixe et virgule flottante. Il est alors possible de comparer les sorties de l'opérateur de décision dans le mode de convergence asymptotique ce qui permet d'obtenir la probabilité d'erreur de décision par simulation Monte Carlo. Par ailleurs, nous avons programmé en langage C l'algorithme de Newton-Raphson afin de solutionner le système non linéaire (3.9) décrit précédemment, et ainsi d'obtenir quasi-analytiquement la probabilité d'erreur de décision dans le mode de convergence asymptotique. La figure 3.3 montre la probabilité d'erreur de décision obtenue analytiquement par notre modèle ainsi par ainsi que celle obtenue par simulations dans le cas particulier d'une constellation de type QPSK et d'un filtre H à retour de décision constitué de 4 coefficients $\{h_1, h_2, h_3, h_4\}$.

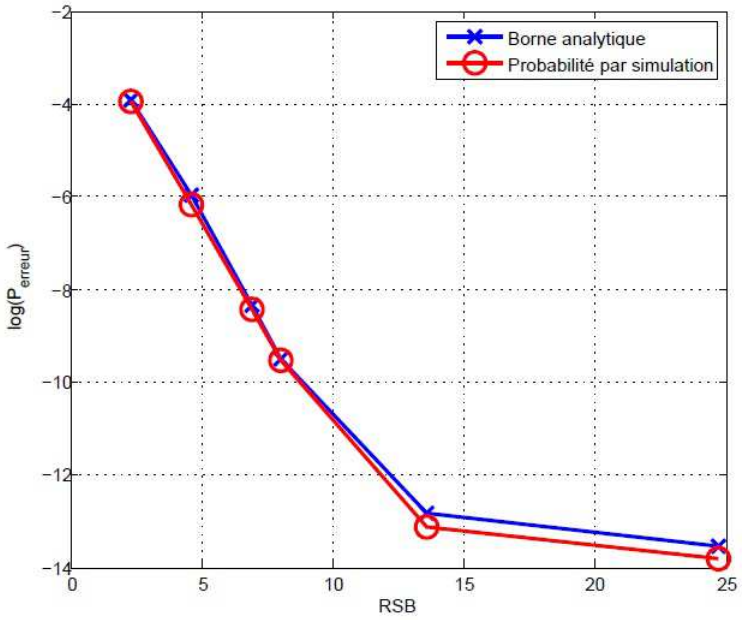

FIGURE 3.3 – Probabilité d'erreur de décision pour une constellation QPSK

Nous remarquons que les résultats analytiques correspondent à ceux obtenus par simulation avec une marge d'erreur inférieure à 1%. Nous pouvons donc conclure que le modèle analytique proposé permet de prendre en compte la cascade d'opérateurs de décision induits par le caractère récursif du système

3.2.2 Borne supérieure de la probabilité d'erreur

Lorsque le calcul de la probabilité d'erreur de décision devient trop complexe d'un point de vue purement calculatoire, une autre alternative fondée sur une borne supérieure de la probabilité d'erreur de décision peut être envisagée. Nous présentons dans cette section cette deuxième approche pour évaluer la précision en virgule fixe des systèmes constitués d'une structure récursive intégrant un ou plusieurs opérateurs non lisses de décision. Soit e_k l'erreur de décision causée par la conversion en format virgule fixe à l'instant k. Chaque erreur e_{k-j} prend une valeur d'un ensemble de D valeurs distinctes. Si seulement $K < \infty$ erreurs passées affectent la décision en cours, la séquence d'erreur $(e_{k-1}, ..., e_{k-K})$ est par conséquent une des D^K valeurs distinctes. Si chacune des D^K séquences est assignée à un état, les séquences d'états résultantes constituent une chaîne de Markov. Nous considérons ci-après une alternative pour définir l'affectation des états du système. Pour cela, soit V un entier arbitraire, nous définissons alors les $(V + 1)$ états du système suivant :

$$\begin{aligned} E_m &= \{e_{k-1}, e_{k-2}, \ldots, |e_{k-j} = 0, 1 \leq j \leq m, e_{k-m-1} \neq 0\}, \qquad 0 \leq m \leq V-1 \\ E_V &= \{e_{k-1}, e_{k-2}, \ldots, |e_{k-j} = 0, 1 \leq j \leq V\}. \end{aligned} \tag{3.20}$$

Par conséquent, E_m consiste à représenter toutes les séquences d'erreurs constituées de m valeurs nulles suivies d'une unique valeur d'erreur non nulle (à part pour le cas de E_V, pour lequel e_{k-V-1} peut également être nulle). L'ensemble de ces $(V+1)$ états peut être illustré par le tableau suivant :

Etat	Séquence d'erreur (E=Erreur, X=Peu importe, 0=Pas d'erreur)
E_0	$EXX...$
E_1	$0EXX...$
E_2	$00EXX...$
.	
.	
.	
E_{V-1}	00...0EXX...(M-1 zéros)
E_V	00...00XX...(M zéros)

TABLE 3.1 – État des séquences d'erreurs

Dans une seconde étape, nous définissons un processus aléatoire S_k qui prend une des valeurs entières $\{0, 1, ..., V\}$. Supposons que l'état du système soit égal à E_j lorsque le signal x_k est reçu et définissons alors $S_k = j$. S_k est la séquence des états du système. La probabilité que l'état du système soit E_m lorsque le signal x_k est reçu est la probabilité que $S_k = m$ et elle est définie par la somme des probabilités conditionnelles sous contrainte de l'état du système à l'itération $k-1$.

$$p(S_k = m) = \sum_{i=0}^{V} p(S_k = m | S_{k-1} = i) p(S_{k-1} = i). \tag{3.21}$$

Notons la probabilité $p(S_k = i) = p_i$, par conséquent, p_m s'écrit par la relation suivante :

$$p_m = \sum_{i=0}^{V} p(S_k = m | S_{k-1} = i) p_i, \qquad 0 \leq m. \leq V \tag{3.22}$$

Nous pouvons remarquer que l'état $S_k = j$ ne peut être atteint qu'à partir de l'état $S_{k-1} = j-1, 1 \leq m \leq M$. Par conséquent, la plupart des probabilités de transition dans l'équation (3.22) sont nulles. De ce fait, nous définissons la probabilité α_m d'une décision correcte étant donné que l'état du système est $S_k = m$, soit encore :

$$\alpha_m = p\{e_k = 0 | S_k = m\}. \tag{3.23}$$

Par conséquent, la probabilité que l'état du système soit m est simplifiée par :

$$p_0 = \sum_{m=0}^{V} (1 - \alpha_m) p_m. \tag{3.24}$$

$$p_m = \alpha_{m-1} p_{m-1}, 1 \leq m \leq V-1. \tag{3.25}$$

$$p_V = \alpha_{V-1} p_{V-1} + \alpha_V p_V. \tag{3.26}$$

Les probabilités $p_m, m = 1, ..., V-1$ et p_M peuvent être exprimées à partir de p_0 suivant :

$$p_m = p_0 \prod_{i=0}^{m-1} \alpha_i, 1 \leq m \leq V-1 \tag{3.27}$$

et :

$$p_V = p_0(1-\alpha_V)^{-1}\prod_{i=0}^{V-1}\alpha_i. \tag{3.28}$$

En tenant compte du fait que $\sum_{i=0}^{V} p_i = 1$, nous obtenons :

$$p_0 = Q_V^{-1} \tag{3.29}$$

avec :

$$Q_V = 1 + \sum_{i=0}^{V-2}\prod_{m=0}^{i}\alpha_m + (1-\alpha_V)^{-1}\prod_{m=0}^{V-1}\alpha_m. \tag{3.30}$$

Notre but est de déterminer la probabilité d'erreur de décision totale P_E définie par :

$$P_E = p\{e_k \neq 0\} = p_0 = Q_V^{-1}. \tag{3.31}$$

L'équation (3.29) nous donne uen expression analytique de cette probabilité d'erreur. Il est possible de calculer les probabilités de transition $\alpha_j, 0 \leq j \leq V$. Par contre, ce calcul nécessite des capacités calculatoires importantes, et ceci d'autant plus que la constellation utilisée possède un haut rendement spectral (i.e. le cardinal de la constellation devient important). Pour solutionner ce problème, nous proposons de déterminer une borne supérieure de la probabilité d'erreur de décision P_E. Pour cela, il est suffisant de borner chacune des probabilités $\alpha_j, j = 0, ..., V-1$, ce qui revient à trouver α_j^* tel que :

$$0 \leq \alpha_j^* \leq \alpha_j \leq 1. \tag{3.32}$$

Pour $V \leq m \leq K$, toutes les probabilités de transition sont égales. Elles sont définies par :

$$\alpha_m = 1 - \epsilon \tag{3.33}$$

où ϵ désigne la probabilité d'erreur en l'absence d'erreurs de décisions passées. Par conséquent la quantité Q_K est donné par :

$$Q_K = 1 + \sum_{i=0}^{V-1}\prod_{m=0}^{i}\alpha_m + \sum_{i=V}^{K-2}\prod_{m=0}^{i}\alpha_m + (1-\alpha_K)^{-1}\prod_{m=0}^{V-1}\alpha_m.\prod_{m=V}^{K-1}\alpha_m \tag{3.34}$$

soit encore :

$$Q_K = 1 + \sum_{i=0}^{V-1}\prod_{m=0}^{i}\alpha_m + \sum_{i=V}^{K-2}(1-\epsilon)^{i-V+1}\prod_{m=0}^{V-1}\alpha_m + \epsilon^{-1}(1-\epsilon)^{K-V}\prod_{m=0}^{V-1}\alpha_m. \tag{3.35}$$

et :

$$Q_V = 1 + \sum_{i=0}^{V-2}\prod_{m=0}^{i}\alpha_m + \epsilon^{-1}\prod_{m=0}^{V-1}\alpha_m. \tag{3.36}$$

Nous avons montré dans [10] que dans le cas d'une constellation de type M-QAM, $\alpha_m \geq \frac{1}{M}$. Par conséquent, en insérant cette relation dans (3.36), nous obtenons un borne inférieure de Q_V donnée par :

$$Q_V \geq 1 + \sum_{i=0}^{V-2}(\frac{1}{M})^{i+1} + \frac{1}{\epsilon}(\frac{1}{M})^V \tag{3.37}$$

$$Q_V \geq \frac{(M-1)(\epsilon M^V + 1) + \epsilon M^2(M^{V-1}-1)}{\epsilon M^V(M-1)}. \tag{3.38}$$

Ce qui engendre une majoration de la probabilité d'erreur de décision P_E qui est égale à Q_V^{-1} par :

$$P_E \leq \frac{\epsilon M^V (M-1)}{(M-1)(\epsilon M^V + 1) + \epsilon M^2 (M^{V-1} - 1)}. \tag{3.39}$$

Ce résultat constitue un résultat très intéressant car il permet de déterminer une borne supérieure de la probabilité d'erreur de décision en fonction du nombre M d'états de la constellation QAM, et également en fontion du paramètre ϵ qui exprime le format de quantification sélectionné. Nous proposons de tester ce modèle analytique en considérant une application fondamentale de tout récepteur numérique moderne : l'égaliseur à retour de décision ou DFE (pour *Decision Feedback Equalizer*) dont le fonctionnement sera explicité dans la section ci-après.

3.3 Évaluation de la précision de l'égaliseur à retour de décision

Les canaux de transmission réels, tels que les canaux sans-fil de type 4G ou bien encore la transmission sur fibre optique à 400 Gbit/s, sont caractérisés par une propagation de type multi-trajet pouvant introduire un niveau d'interférence entre symboles (ISI) important ce qui peut, dans certains cas, être le principal facteur limitatif du système de transmission considéré. Pour combattre cette limitation, il est possible d'intégrer au sein du récepteur un égaliseur à retour de décision (ou DFE). Cette technique a attiré une attention considérable en raison de sa capacité à supprimer l'interférence entre symbole postcurseur par la partie récursive (ou *feedback*) de l'égaliseur. De plus, étant donné que seule la partie directe (ou *feedfoward*) est perturbée par le bruit introduit par le canal de transmission, le compromis bruit/performances ne se pose pas dans la partie récursive ce qui constitue un avantage certain. Le principe dans le fonctionnement d'un tel égaliseur est présenté dans la section suivante.

3.3.1 Présentation de l'algorithme DFE

Du fait de leurs nombreux avantages, les égaliseurs à retour de décision (DFE) sont couramment utilisés afin d'éliminer dans les signaux de communication la distorsion induite par les canaux de propagation. En effet, même lorsque l'interférence entre symbole engendrée par le canal est sévère et que le niveau de bruit est important, il est possible d'optimiser les paramètres de l'égaliseur afin qu'il atteigne des performances optimales, i.e. une erreur quadratique moyenne faible aux bornes du circuit de décision. Par ailleurs, du point de vue complexité matérielle, cette technique s'avère très compétitive par rapport à des solutions alternatives telles que par exemple le récepteur à maximum de vraisemblance.

Le bloc-diagramme d'un égaliseur à retour de décision est présenté à la figure 3.4. Cet égaliseur est constitué par deux filtres : le filtre *forward* et le filtre *feedback* (retour d'information). Les deux filtres sont généralement implémentés à un rythme symbole cependant le filtre *forward* peut également être implémenté à un rythme fractionnaire (par rapport au rythme symbole) ce qui procure une immunité du récepteur par rapport au choix de l'instant optimal d'échantillonnage. Ce filtre est un filtre transverse qui prend en entrée la sortie du filtre récepteur et de l'échantillonneur. Par conséquent, il ressemble d'une certaine façon à un filtre linéaire.

Le filtre *feedback* prend en entrée les sorties passées de l'opérateur de décision. Puisque ce filtre utilise les décisions passées pour générer sa sortie, il doit être strictement causal. Le filtre *forward* enlève une partie des interférences ISI du signal reçu, mais en contrepartie laisse une partie des interférences ISI dans le signal (généralement, il s'agit d'interférence intersymbole postcurseur). Le filtre *feedback* estime l'interférence ISI résiduelle des décisions passées et il la soustrait de la sortie du filtre *forward* comme montré dans la figure 3.4.

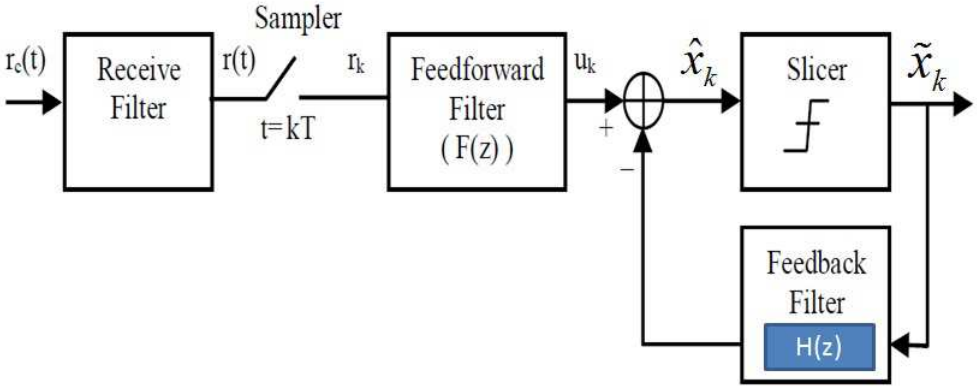

FIGURE 3.4 – Bloc-diagramme d'un égaliseur à retour de décision

La solution à retour de décision donne généralement de meilleures performances que l'égaliseur linéaire (i.e. uniquement la partie *forward*) et ceci pour une complexité additionnelle acceptable. La faible amélioration du bruit du DFE provient du fait qu'en supposant qu'il n'y a pas d'erreur de décision, l'opérateur de décision enlève tout le bruit présent dans le signal [52].

3.3.2 Résultat pour le cas non adaptatif

Dans cette section, nous utilisons l'approche exposée dans le paragraphe 3.2.2 reposant sur le calcul d'une borne supérieure de la probabilité d'erreur de décision pour évaluer la précision d'un égaliseur non adaptatif à structure fixe comme indiqué dans la figure 3.4. Les signaux reçus sont composés des M symboles de la constellation M-QAM perturbés par un bruit de canal supposé blanc, centré et gaussien. Afin d'évaluer les performances de l'approche proposée, nous faisons varier le format de la représentation en virgule fixe, puis, pour chaque format, nous calculons la probabilité d'erreur de décision ϵ afin d'estimer la borne supérieure. D'autre part nous simulons le DFE en virgule fixe et en virgule flottante afin d'obtenir également la probabilité d'erreur de décision par simulation de type Monte Carlo.

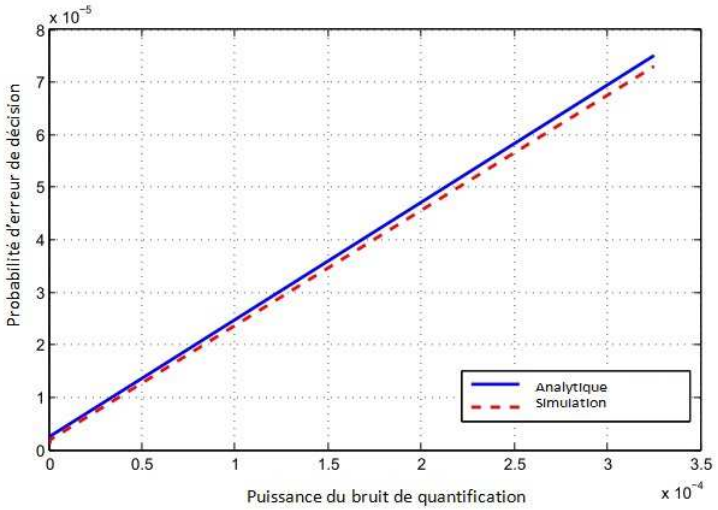

FIGURE 3.5 – Borne supérieure analytique et probabilité d'erreur de décision par simulation

La figure 3.5 montre la probabilité d'erreur de décision estimée par simulation ainsi que celle

obtenue de manière analytique à l'aide de l'approche proposée dans le cas d'une constellation 4-QAM et d'un filtre à retour de décision comportant 4 coefficients. Nous pouvons remarquer que la probabilité obtenue avec l'approche analytique proposée se révèle plus précise lorsque le bruit de quantification possède une faible puissance ce qui est généralement le cas en pratique.

3.3.3 Solution proposée pour le cas adaptatif

Comme le canal de propagation est inconnu, l'implémentation de l'égaliseur à retour de décision DFE doit se faire d'une façon adaptative. Dans un DFE classique l'adaptation se fait par le biais d'une séquence d'apprentissage, ce qui engendre une diminution du débit utile de la transmission. C'est pour cette raison que de nombreux travaux se sont portés sur la recherche de solutions d'égaliseurs ne nécessitant aucune séquence d'apprentissage. On parle alors d'égalisation aveugle. Dans cet esprit, nous nous sommes intéressés à la solution proposée dans [52] où l'égaliseur possède deux modes de fonctionnement : le mode initial où l'égaliseur est formé de la cascade d'un filtre blanchissant purement récursif R, d'un filtre transverse J et d'un correcteur de phase CP. L'originalité du dispositif provient du fait que chaque étage est adapté par rapport à un critère local spécifique ce qui améliore la robustesse du système global ainsi que la rapidité de convergence des coefficients des filtres numériques vers la solution optimale. Une fois que le processus d'égalisation est suffisamment avancé (ce qui se mesure à travers l'erreur quadratique moyenne aux bornes du circuit de décision), la structure du récepteur est modifiée en un égaliseur conventionnel de type DFE pour lequel les coefficients de la partie *backward* sont directement obtenus à partir de ceux du filtre blanchissant R. Le caractère réversible de cette modification apporte au nouvel égaliseur un avantage important qui se manifeste dans le fait qu'il peut apprendre de ses propres décisions sans risque de divergence.

Considérons une séquence de données discrètes $d(k)$ de moyenne nulle et de variance égale à l'unité. Soit $H(z)$ la fonction du transfert du canal de propagation définie par :

$$H(z) = \sum_{l=0}^{N} h(l) z^{-l}. \tag{3.40}$$

Le signal observé est perturbé par un bruit blanc gaussien additif $n(k)$ de variance σ_n^2. Le signal reçu s'exprime de la façon suivante :

$$s(k) = \sum_{l=0}^{N} d(k-l)h(l) + n(k). \tag{3.41}$$

L'égaliseur linéaire MMSE (solution de Wiener) admet comme fonction de transfert :

$$C(z) = \frac{\sigma_d^2 H^*(1/z^*)}{\sigma_d^2 H(z) H^{*\cdot}(1/z^*) + \sigma_n^2} \tag{3.42}$$

Généralement, cet égaliseur est implémenté en utilisant un filtre transverse d'ordre important (idéalement infini) pour éviter les risques d'instabilité. Cependant, l'utilisation d'un filtre récursif (de type IIR) possède des avantages certains. En premier lieu, le choix d'une structure récursive nécessite en général un nombre de paramètres, i.e. de coefficients, plus faible que celui requis par un filtre transverse équivalent. Enfin, cette structure permet de faire évoluer simplement un schéma d'égalisation linéaire vers un schéma d'égalisation non linéaire de type retour de décision. En effet, les coefficients du filtre IIR blanchissant sont estimés durant la première phase de l'algorithme et correspondent aux coefficients de prédiction linéaire arrière (ou *backward*) du signal reçu. Ensuite, dans la seconde phase de l'algorithme, ces coefficients sont tout simplement utilisés pour initialiser les coefficients du filtre à retour de décision. Considérons la relations (3.42), le dénominateur de $C(z)$ n'est autre que la densité spectrale de puissance (d.s.p) $S_s(z)$ du signal observé $s(k)$. Par conséquent, si N est le nombre de zéros de la fonction de transfert du canal $H(z)$, alors $S_s(z)$ possède $2N$ racines, N d'entre elles étant à l'intérieur du cercle unité et les N autres étant à l'extérieur. La DSP $S_s(z)$ peut ainsi se factoriser comme suit :

$$S_s(z) = KG(z)G^*(1/z^*). \tag{3.43}$$

avec $G(z)$ est un polynôme à phase minimale et K une constante réelle positive, on a alors :

$$G(z) = \prod_{k=1}^{N}(1 - z_k z^{-1}) \qquad avec \qquad |z_k| < 1. \tag{3.44}$$

Ainsi, la fonction de transfert de l'égaliseur est la suivante :

$$C(z) = \frac{1}{G(z)} \frac{\sigma_d^2}{K} \frac{H^*(\frac{1}{z^*})}{G^*(\frac{1}{z^*})}. \tag{3.45}$$

Le filtre $\frac{1}{G(z)}$ peut alors être implémenté comme un filtre récursif stable. Ce filtre est également le filtre blanchissant les observations parce que la densité spectrale de puissance de sa sortie est égale à A. Par conséquent, l'égaliseur linéaire optimal est implémenté tel qu'indiqué sur la figure 3.6, où $A(z) = G(z) - 1$ et $B(z) = \frac{\sigma_d^2}{K} \frac{H^*(\frac{1}{z^*})}{G^*(\frac{1}{z^*})}$.

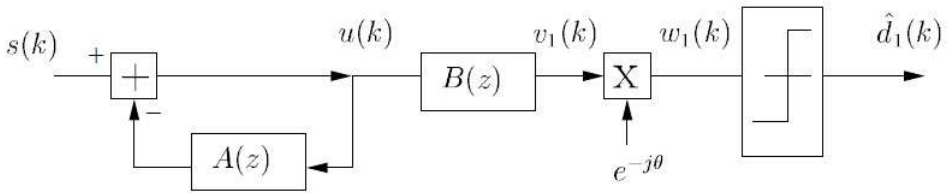

FIGURE 3.6 – Structure de l'égaliseur : mode initial

Si nous comparons l'égaliseur linéaire optimal avec le DFE conventionnel (figure 3.7), nous pouvons voir que le passage de l'un à l'autre se fait simplement par un changement de structure : inversion de la position des filtres $A(z)$ et $B(z)$.

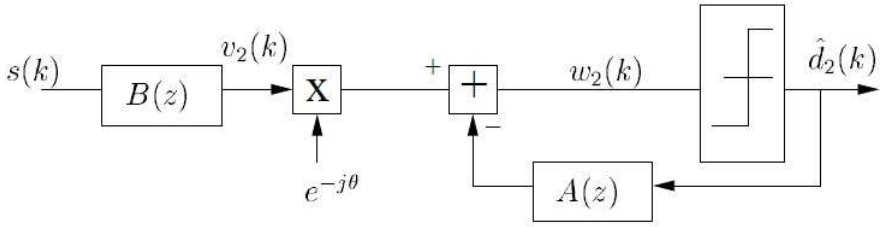

FIGURE 3.7 – Structure de l'égaliseur : mode final

Dans le contexte aveugle, il suffit de trouver les critères permettant de trouver une solution proche de celle du MMSE. La solution de mise en œuvre consiste à mettre en place un algorithme itératif. La détermination du filtre $A(z) = \sum_{k=1}^{L_A} a_k z^{-k}$ demande de minimiser :

$$J_1(A) = E[|u(k)|^2]. \tag{3.46}$$

Avec $A = (a_1, a_2, ...a_{L_A})^T$. L'algorithme itératif qui donne la solution désirée s'écrit alors à l'itération k par :

$$u(k) = s(k) - A^{(k-1)T} U_{L_A}(k-1), \tag{3.47}$$

$$A^k = A^{(k-1)} + v_A u(k) U_{L_A}^*(k-1). \tag{3.48}$$

où $A^0 = [0, 0, ..0]^T$ et $U_{L_A}(k-1) = (u(k-1), ..., u(k-L_A))^T$ et v_A est un pas positif correctement choisi et assurant la convergence de l'algorithme.

Plusieurs algorithmes aveugles de déconvolution permettent de déterminer le filtre $B(z) = \frac{\sigma_d^2}{K}\frac{H^*(\frac{1}{z^*})}{G^*(\frac{1}{z^*})}$. Le critère de Godard [47] permet de déterminer un égaliseur $B(z)$ de type zero forcing dans un contexte non bruité. Si le bruit n'est pas très fort, le filtre $B(z)$ résultant d'une minimisation du critère de Godard est proche de la solution MMSE souhaitée. L'algorithme itératif qui minimise le critère de Godard est le suivant :

$$v_1(k) = B^{(k-1)T} U_{L_B+1}(k), \tag{3.49}$$

$$B^{(k)} = B^{(k-1)} + v_B v_1(k)(1 - |v_1(k)|^2) U^*_{L_B+1}(k). \tag{3.50}$$

Où $B^0 = [0, ..., 01, 0, ..., 0]^T$ et v_B un pas positif bien choisi. Nous obtenons à la sortie du CMA (*constant modulus algorithm*) une estimée des symboles à un retard près et une constante complexe près. Donc, il est nécessaire de mettre en place une méthode de correction de phase. Le critère retenu est :

$$J_3(\theta) = E[|v_1(k)\exp(-j\theta) - \hat{d}_1(k)|^2]. \tag{3.51}$$

Où $\hat{d}_1(k)$ est le k^{eme} symbole décidé. Dès que les filtres $B(z)$ et $A(z)$ sont proches de leur valeur finale (lorsque l'erreur quadratique estimée est inférieure à un seuil), le changement de structure peut être effectué. Les filtres $B(z)$, $A(z)$ et θ sont alors adaptés de manière à minimiser le critère (3.51). L'algorithme du gradient stochastique s'écrit :

$$A^{(k)} = A^{(k-1)} + v_A(\hat{d}_2(k) - w_2(k))\hat{D}^*(k), \tag{3.52}$$

$$B^{(k)} = B^{(k-1)} + v_B(\hat{d}_2(k) - w_2(k))\exp(j\theta(k-1))T^*(k), \tag{3.53}$$

$$\theta(k) = \theta(k-1) + v_\theta(\epsilon(k) + \beta\sum_{i=1}^{k}\epsilon(i)). \tag{3.54}$$

Où

$$\begin{aligned} y(k) &= (B^{(k-1)T}T(k))\exp(j\theta(k-1)), \\ T(k) &= [s(k), ..., s(k-L_B)]^T, \\ \hat{D}(k) &= [\hat{d}_2(k-1), ...\hat{d}_2(k-L_A)]^T, \\ \epsilon(k) &= Im[w_2(k)(\hat{d}_2(k) - w_2(k))^*], \end{aligned} \tag{3.55}$$

et v_θ est un pas positif bien choisi assurant la convergence de l'algorithme, β est un paramètre positif et $\theta(0) = 0$.

Cet algorithme possède l'avantage majeur de posséder une très bonne réactivité et ceci pour un faible coût calculatoire. Son inconvénient est le temps de convergence. Il est également possible d'adapter le mode de fonctionnement de l'algorithme à un mode *burst* adapté aux systèmes de transmission (tels que le GSM, la 3G, le LTE) où l'information utile est transmise sous forme de slots. Le principe général de ce mode *burst* repose sur le constat que les paramètres A, B et θ sont plus proches de leurs valeurs optimales à la fin d'un bloc plutôt que au début d'un bloc. Il est alors possible d'itérer la procédure sur un même bloc en initialisant l'algorithme avec les valeurs obtenues en fin de bloc lors de la précédente itération. Ceci permet de réduire nettement le temps de convergence de l'algorithme moyennant une perte de réactivité. Bien évidemment, dans ce fonctionnement en mode *burst*, il est nécessaire que le canal de propagation soit stationnaire sur la durée du bloc (ce qui est généralement le cas car la durée d'un bloc est choisie comme étant inférieure au temps de cohérence du canal de propagation).

Afin de pourvoir évaluer la précision d'un tel égaliseur adaptatif, nous simulons l'algorithme DFE adaptatif en utilisant l'algorithme LMS pour mettre à jour les coefficients des filtres *feedback* et *forward*. Dans un second temps, nous remarquons que la variation des coefficients des filtres est très petite lorsque le DFE converge voire même quasiment nulle. La figure 3.8 montre la variation des coefficients des filtres *forward* et *feedback* au cours du temps.

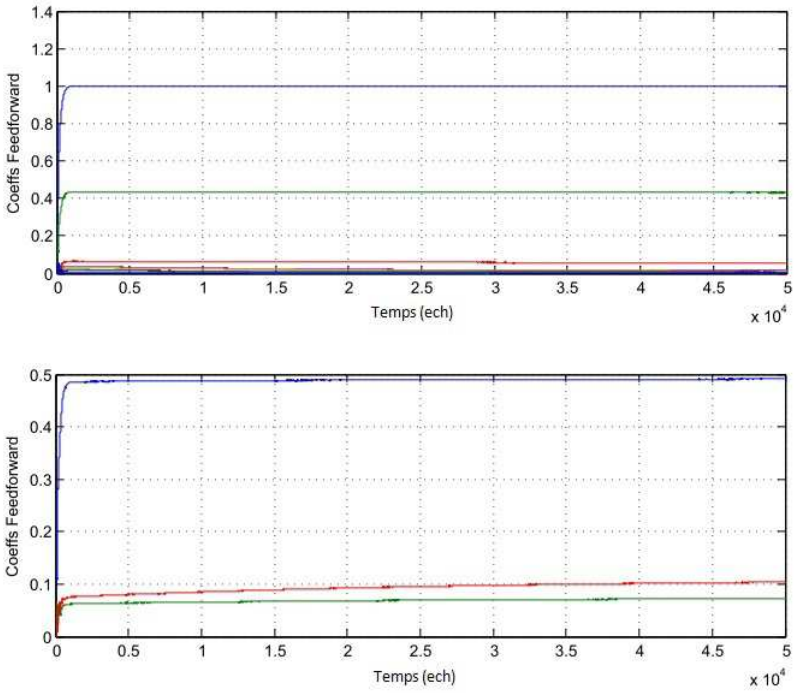

FIGURE 3.8 – Variation des coefficients des filtres de l'égaliseur

Par conséquent, nous prenons les valeurs des coefficients des filtres dans la zone de convergence et nous les utilisons pour un égaliseur non adaptatif. Ensuite, nous procédons de la même manière qu'un égaliseur non adaptatif en utilisant le modèle analytique de la borne supérieur pour évaluer la probabilité d'erreur de décision dans le régime de convergence de l'égaliseur DFE.

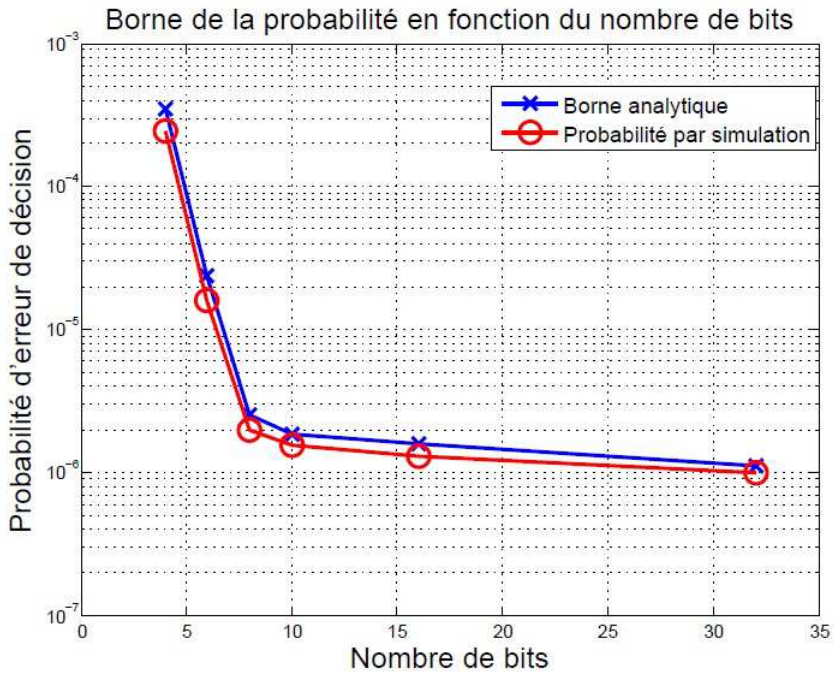

FIGURE 3.9 – Probabilité d'erreur de décision en fonction du nombre de bits

Nous reprenons ci-dessous les mêmes conditions expérimentales que celles utilisées précédemment dans le paragraphe 3.2.2. Afin d'évaluer les performances du modèle proposé, nous faisons

varier le nombre de bits du format de représentation en virgule fixe des paramètres (coefficients, signaux) caractérisant l'égaliseur adaptatif de type DFE. Pour chaque format de représentation en virgule fixe, nous évaluons la borne supérieure de la probabilité d'erreur de décision à partir du modèle analytique proposé ainsi que celle estimée par des simulations de type Monte-Carlo. Ces résultats sont représentés sur la figure 3.9 pour une modulation numérique de type 4-QAM. Nous pouvons constater que le modèle analytique proposé permet d'obtenir une estimation fiable de la probabilité d'erreur de décision sur une large gamme de formats de représentation.

D'après la figure précédente, il est clair que plus la précision est importante (i.e. augmentation du nombre de bits accordés au format en virgule fixe, plus la probabilité d'erreur de décision causée par la perte en précision de la quantification du format en virgule fixe diminue. Cependant, l'augmentation de la précision engendre également des effets sur la consommations des unités de calcul ainsi que sur l'utilisation des ressources matérielles. Par conséquent, le choix d'un format spécifique de représentation en virgule fixe doit résulter d'un compromis entre la précision obtenue et la consommation du circuit électronique. Cet aspect fera l'objet de la prochaine section où nous montrons que les modèles analytiques proposés peuvent être utilisés efficacement afin de sélectionner un format de représentation en virgule fixe optimisant l'implémentation matérielle d'un égaliseur de type DFE sur FPGA.

3.4 Optimisation du format virgule fixe pour l'implémentation matérielle

Dans cette section, l'intérêt du modèle analytique proposé est analysé pour une problématique correspondant à l'implémentation d'un égaliseur de type DFE à coefficients complexes sur FPGA. Nous commençons par présenter la problématique de l'évaluation des coûts de l'implémentation, ensuite nous représentons l'architecture considérée pour l'égaliseur à coefficients complexes et, enfin, nous présentons le choix d'implémentation que nous avons retenu en fonction du compromis entre précision et consommation en énergie et en ressources matérielles.

3.4.1 Présentation de l'architecture

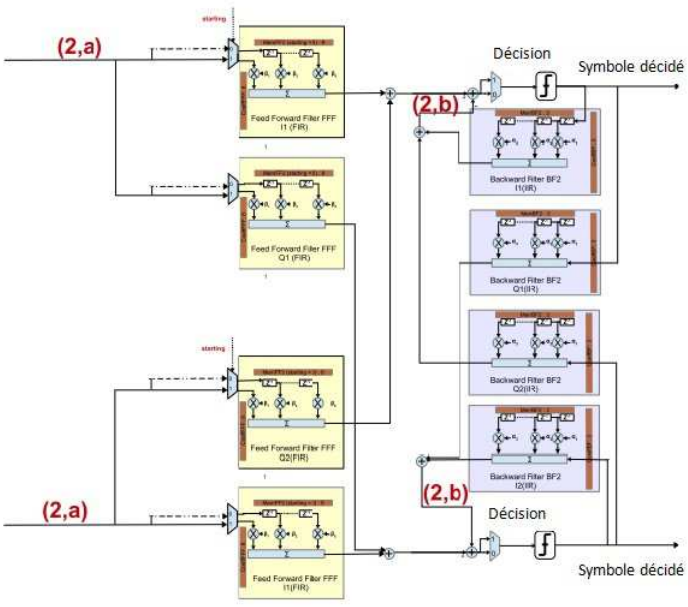

FIGURE 3.10 – Architecture d'un égaliseur DFE à coefficients complexes

En raison du grand nombre d'applications qui l'utilisent, nous avons choisi d'étudier le cas de l'égaliseur DFE à coefficients complexes composé de deux voies : la voie réelle (ou *en phase*) et la voie imaginaire (ou *en quadrature*). Chacune de ces voies est assimilée à une DFE réel mais, additionnellement, des filtres croisés sont insérés entre voies en phase et quadrature ce qui donne lieu à une architecture dite en papillon (ou *butterfly*) comme représenté à la figure 3.10.

Un modèle matériel de l'égaliseur a été développé en utilisant le langage de description matériel HDL(*Hardware Description Language*) et synthétisé en utilisant l'outil de synthèse de Xilinx (ISE Xilinx). Les simulations du comportement du modèle et la post-synthèse ont été effectuées en utilisant ModelSim. Le schéma d'implémentation de l'égaliseur DFE conçu est donné par la figure 3.11.

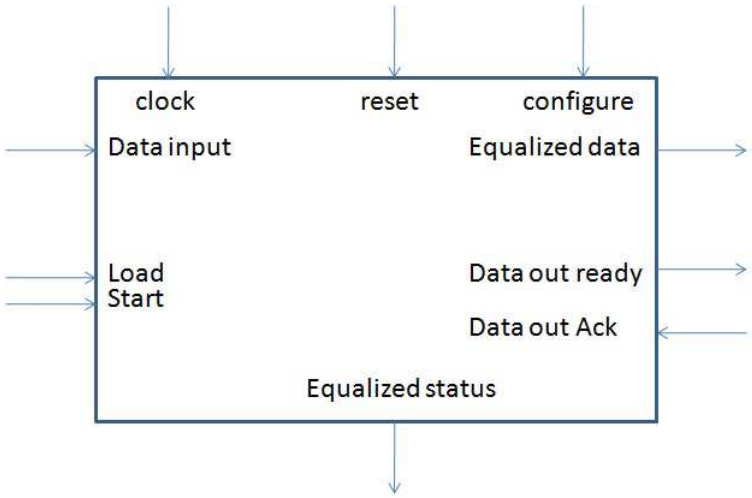

FIGURE 3.11 – Diagramme bloc de l'égaliseur DFE implémenté

L'égaliseur consiste en une horloge globale et des entrées *Reset* synchrones. Les entrées sont introduites dans l'égaliseur en utilisant un signal de contrôle de chargement (*Load Control Signal*). Le processus d'égalisation est initialisé par le signal début (*Start Signal*). Après la fin de l'égalisation, les données égalisées (*Equalized Data*) peuvent être obtenues lorsque le signal sortie prête (*DataOut Ready Signal*) est positionné. La réception des données peut être connue par le biais du signal d'acquittement (*DataOut Ack Signal*) pour recevoir le prochain bit. Le port état de l'égaliseur (*Equalized Status*) indique la progression (*Active/Idle*) de l'égaliseur. Il convient de préciser que les entrées sont chargées en série dans l'égaliseur. Également, les données égalisées sont lues bit par bit en série. Cette technique est utilisée en raison du nombre limité des ports entrées sorties disponibles dans le FPGA. Elle fournit également des flexibilités pour implémenter l'égaliseur DFE avec des longueurs de précision variables sans modifier le port de configuration.

3.4.2 Implémentation sur FPGA

Afin de démontrer de l'intérêt des méthodes analytiques proposées dans le cadre de ce travail, nous avons implémenté l'égaliseur DFE à coefficients complexes sur une carte FPGA de type Xilinx Vertex 5 (ML550). Un environnement de test a été développé afin de tester l'égaliseur implémenté. Cette procédure du test est illustrée par la figure 3.12.

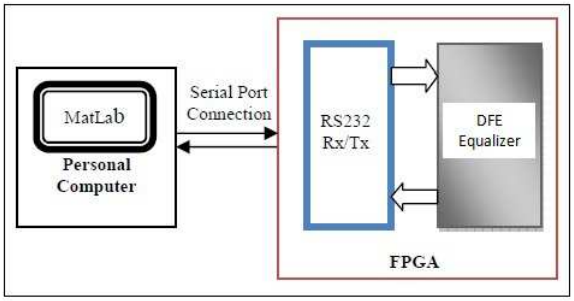

FIGURE 3.12 – Bloc diagramme de la procédure du test sur FPGA

Un module de communication RS232 est embarqué dans le FPGA avec le module d'égaliseur DFE pour faire l'interfaçage avec le port RS232 de l'ordinateur. Pour assurer la communication entre le FPGA et l'ordinateur, nous utilisons le logiciel @Matlab. Un driver du port de communication série a été développé en utilisant la programmation C et exécuté dans l'environnement Matlab. Une vitesse de transmission maximale de l'ordre de 115200 kbps est utilisée pour la communication des données en série. Les données en entrée sont générées avec Matlab et envoyées au FPGA avec les signaux de contrôle appropriés. Après le processus d'égalisation, les données égalisées reçues à travers le même port série sont utilisées pour analyser les performances non seulement de l'égaliseur, mais aussi la précision des données en format virgule fixe. L'architecture considérée est composée de 4 filtres *forward* et 4 filtres *feedback*. Il y a deux filtres en phase et deux filtres en quadrature pour chaque type de filtre. Par conséquent le signal reçu est décomposé en deux signaux (réel et imaginaire). Chaque signal est filtré par les deux filtres *forward* en phase et en quadrature. Ensuite, la partie réelle est filtrée par le filtre *forward* en phase et ajoutée à la partie imaginaire filtrée par le filtre *forward* en quadrature. La même procédure est faite pour la partie imaginaire. Après cela, chaque voie est ajoutée au symbole décidé filtré par les deux filtres *feedback* en phase et en quadrature. L'architecture de chaque filtre est donnée par la figure 3.13

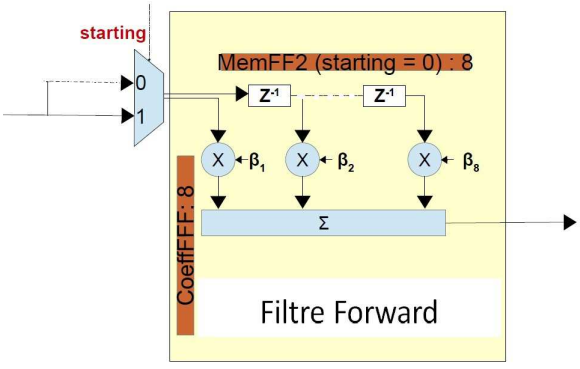

FIGURE 3.13 – Architecture du filtre *forward*

Chaque filtre est composé par deux blocs mémoire de même longueur l : l'un pour stocker les coefficients du filtre et le second pour mémoriser les données en entrée. A l'entrée du filtre, une horloge est utilisée afin d'assurer la synchronisation. Le flot de données en entrée est tout d'abord mémorisé au sein de l'opérateur de retard élémentaire. Chaque valeur est ensuite multipliée par le biais d'un multiplieur avec le coefficient correspondant qui est stocké dans la mémoire des coefficients. Finalement, l'ensemble des valeurs pondérées par les coefficients du filtre est sommé à l'aide d'un additionneur. Pour le filtre *feedback*, nous avons également considéré une architecture identique à celle du filtre *forward*.

Au sein de l'émetteur et du récepteur, des filtres en cosinus surélevé sont utilisés afin de satisfaire à la condition de Nyquist assurant la nullité de l'interférence entre symboles. Le canal de propagation entre T_x et R_x correspond à un canal de type multi-trajets. Une fois que l'égaliseur a atteint son régime asymptotique de convergence, nous prenons les valeurs des coefficients des filtres et les stockons dans les mémoires de nos filtres. Ensuite, nous implémentons notre architecture en utilisant @Matlab pour une constellation de type M-QAM. La figure 3.14 montre que l'égaliseur implémenté effectue bien l'égalisation. Nous commençons par initialiser les mémoires des filtres par des zéros. Ensuite, nous mettons à jour la mémoire des filtres pour chaque itération. Nous avons utilisé 8 coefficients pour le filtre Forward et 4 coefficients pour le filtre Backward avec un facteur de sur-échantillonnage égal à 8.

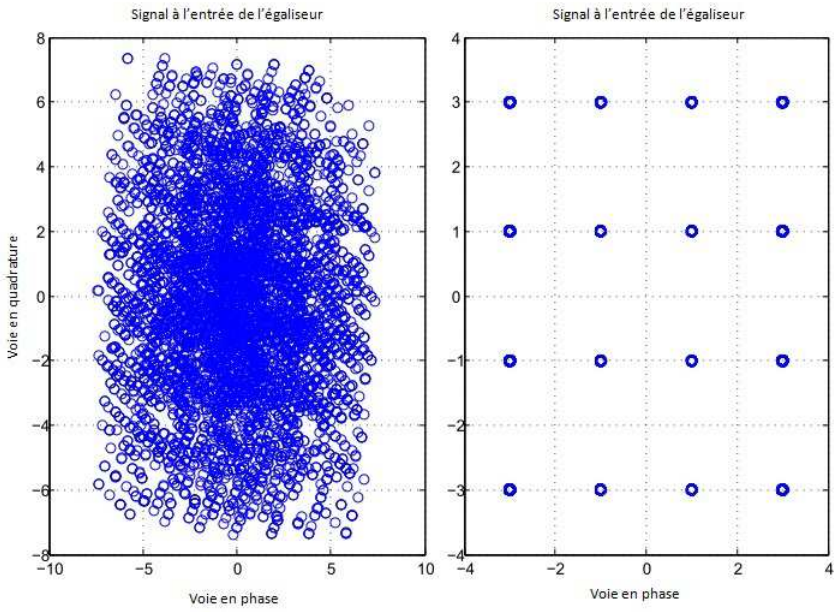

FIGURE 3.14 – Entrée et sortie de l'égaliseur

3.4.3 Les contraintes du choix du format de représentation

Une fois l'architecture générale définie, les données sont ensuite converties en un format de représentation en virgule fixe et il est possible d'estimer la borne de la probabilité d'erreur de décision causée par l'opération de quantification en virgule fixe. Les données en format virgule fixe sont alors utilisées pour estimer les ressources consommées pour le format de précision considéré. En faisant varier la précision des données présentes en entrée du filtre *forward* ainsi que celles de la sortie du filtre *backward* au sein de l'architecture de la figure 3.10, il est alors possible d'estimer la borne supérieure de la probabilité d'erreur de décision en utilisant le modèle analytique décrit précédemment. Pour chaque format de précision considéré, les ressources consommées par l'implémentation FPGA sont également estimées en parallèle à l'aide des outils Xilinx. Pour les tests, nous avons choisi de fixer la partie entière du format des données en virgule fixe à 2 bits. Pour faire varier le format de la précision, le nombre n_a de bits de la partie fractionnaire des entrées du filtre *forward* ainsi que le nombre n_b de la partie fractionnaire de la sortie du filtre *feedback* varient entre 4 et 16 bits. Pour chaque configuration ainsi obtenue définie par le couple (n_a, n_b), nous calculons la borne supérieure de la probabilité d'erreur de décision ainsi que les ressources consommées par le FPGA.

La figure 3.15 est une représentation logarithmique en 3D de la probabilité d'erreur de décision causée par la conversion du format en virgule flottante vers l'arithmétique virgule fixe exprimée en fonction du couple (n_a, n_b) exprimant la précision des données en entrée du filtre *forward* ainsi qu'en sortie du filtre *feedback*. Cette même courbe peut également être projetée sur le plan 2D

(n_a, n_b) en utilisant une échelle de couleur pour exprimer la valeur de la probabilité d'erreur de décision (cf. figure 3.20).

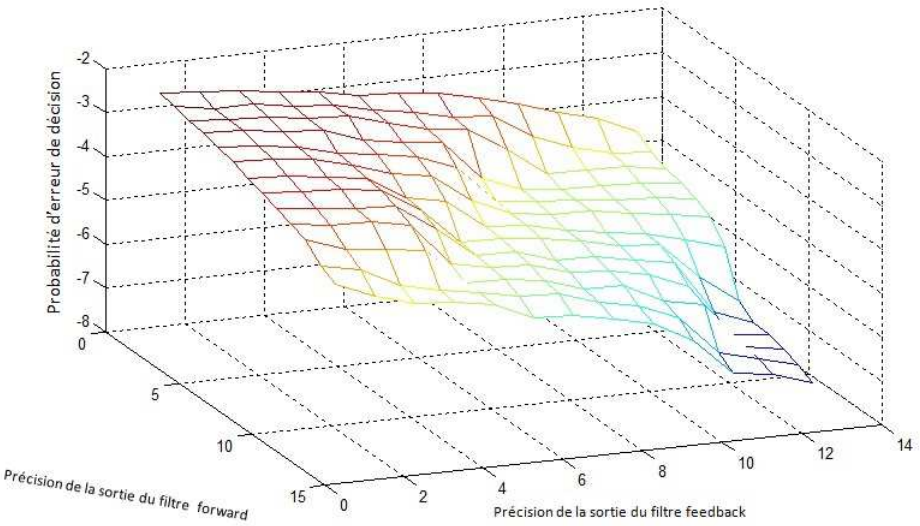

FIGURE 3.15 – Probabilité d'erreur de décision en fonction des deux précisions

Bien évidemment, nous constatons que la probabilité d'erreur de décision diminue lorsque la précision augmente. De plus, nous pouvons trouver plusieurs couples de précision qui donnent approximativement la même probabilité d'erreur de décision. Nous obtenons ainsi un ensemble de couples de précision (n_a, n_b) qui partagent approximativement la même probabilité d'erreur de décision. Sur la figure 3.15, ceci se manifeste par des régions qui possèdent la même couleur. Afin de faire un choix entre les couples de précisions (n_a, n_b) qui appartiennent à un même ensemble de probabilité d'erreur de décision, il est alors intéressant de pouvoir estimer les ressources consommées par le FPGA en termes de nombre de *slices* occupés, de *slice luts* ou *slices* de type Flip Flop.

Nous fixons un critère d'approximation et nous cherchons toutes les précisions qui possèdent approximativement la même borne supérieure de probabilité d'erreur de décision à un facteur prêt. En d'autres termes, nous choisissons une probabilité d'erreur de décision P_0 et un facteur d'approximation α, puis nous cherchons les formats en virgule fixe dont les probabilités d'erreur de décision P vérifient le critère :

$$\frac{|P - P_0|}{P_0} \leq \alpha. \tag{3.56}$$

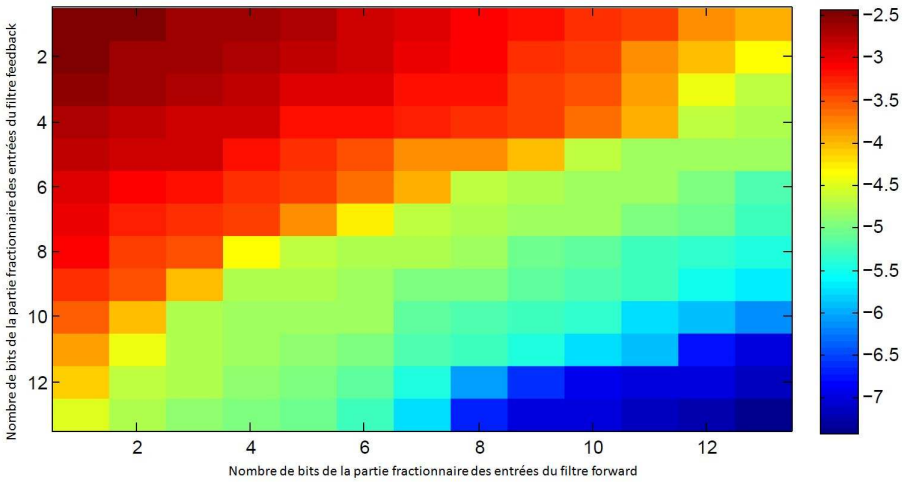

FIGURE 3.16 – Probabilité d'erreur de décision

Nous représentons sur les figures 3.17 et 3.18 le nombre de *slice lut* et *slice registre* occupés pour un ensemble de probabilité d'erreur de décision qui vérifie ce critère pour des paramètres : $P_0 = 10^{-3}$ et $\alpha = 0.2$.

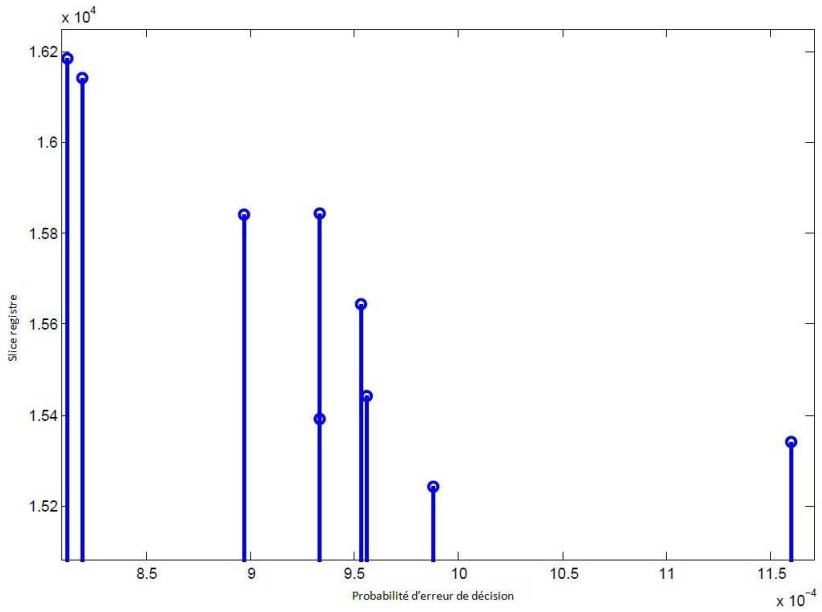

FIGURE 3.17 – Nombre de *slice registres* nécessaires pour obtenir une probabilité d'erreur de décision de $P_0 = 10^{-3}$ avec une précision relative de 0.2.

Par conséquent, nous devons choisir le format de données qui consomme le minimum de ressources sur le FPGA.

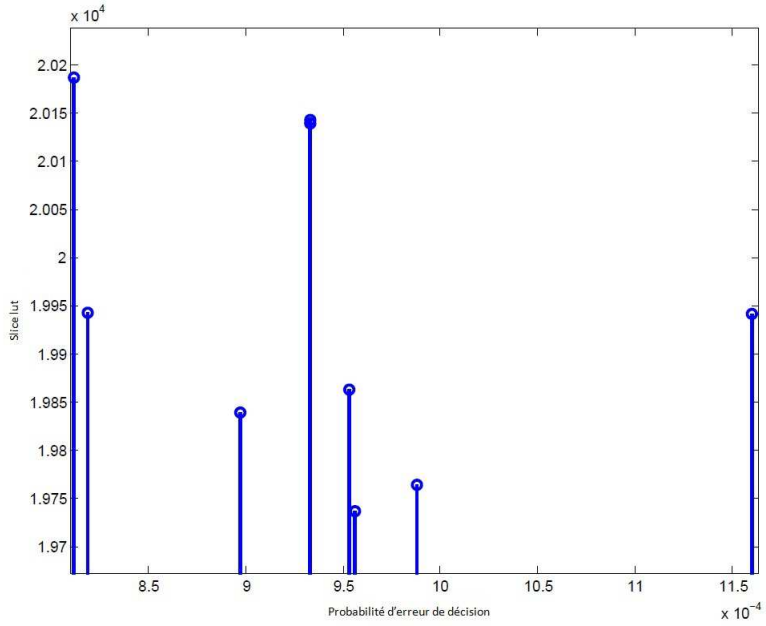

FIGURE 3.18 – Nombre de *slice lut* nécessaires pour obtenir une probabilité d'erreur de décision de $P_0 = 10^{-3}$ avec une précision relative de 0.2.

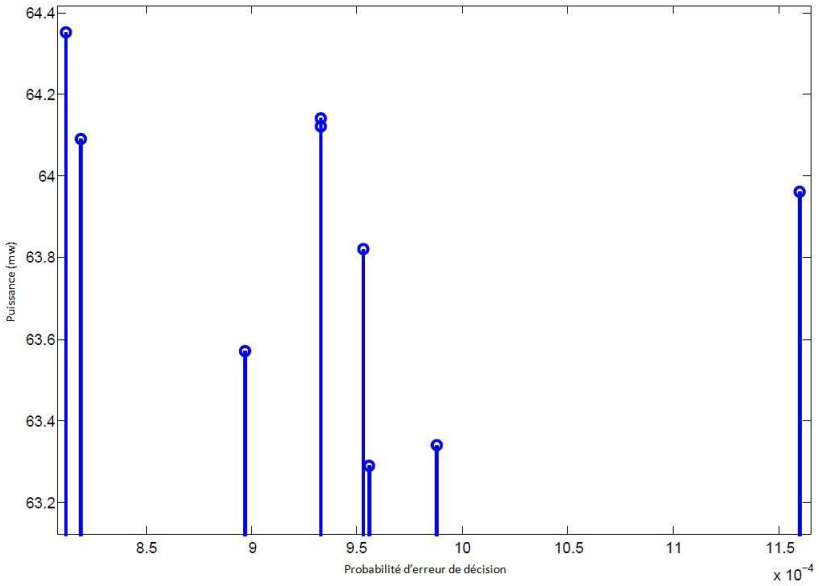

FIGURE 3.19 – Consommation de puissance

Un autre aspect intéressant est la consommation de la puissance ou d'énergie. C'est pour cette raison que nous avons choisi de travailler avec la carte ML550 car elle permet de mesurer la puissance consommée. En effet, cette carte possède une résistance Kelvin qui permet de mesurer le courant circulant à travers le cœur du FPGA. La valeur de cette résistance est suffisamment petite (quelque milli-ohm) pour conduire à une perturbation minimale de l'alimentation. Les variations de la tension à travers la résistance (représentant la variation du courant) sont très petites (quelques mV) et il est difficile de les mesurer. Par conséquent, afin de faciliter les mesures, un amplificateur de tension (le INA333 de chez Texas Instrument) est inséré aux bornes de la résistance (cf. figure 3.20) afin de faciliter la mesure. Cet amplificateur possède un gain de 100

pour une bande passante de $3.5kHz$.

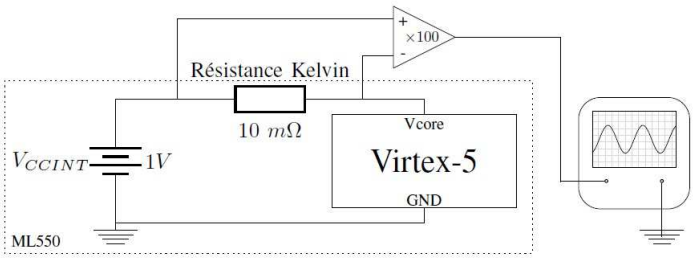

FIGURE 3.20 – Dispositif pour la mesure de la puissance

Si l'on considère une résistance de mesure de courant de $R = 10mW$ sur la carte ML550 et un gain $G = 100$ pour l'amplificateur, nous obtenons une relation directe entre la tension et le courant circulant dans la résistance :

$$U = R * I \tag{3.57}$$

$$U_{ampl} = R * I * G = R * I * 100 = 0.01 * I * 100 = I. \tag{3.58}$$

La tension d'alimentation du FPGA est 1V et la variation de cette tension est inférieure à $0.05V$ (par mesure). La puissance est directement liée à la tension de l'amplificateur.

$$P = U_{ampl} * V \approx U_{ampl}. \tag{3.59}$$

La dernière étape est l'analyse des résultats obtenus en fonction de la précision choisie. A ce stade, nous avons tous les éléments pour analyser l'influence de la précision choisie sur la consommation en énergie et en ressources de l'algorithme. Il suffit alors de mesurer cette consommation pour un ensemble de formats de représentation des données en virgule fixe dont la borne supérieure de probabilité d'erreur de décision satisfait à la condition (3.56). Nous souhaitons pouvoir déterminer le format de données qui engendre une consommation minimale de puissance et de ressources sur le FPGA. A partir de la figure 3.19, nous pouvons constater que ce minimum est atteint pour une consommation d'environ $63.25mW$ ce qui correspond à une probabilité d'erreur de décision de 9.5710^{-4}. Nous constatons sur la figure 3.18 que le format optimal de données précédent (format A) est également celui qui fournit la consommation la plus faible de slice LUT par rapport à toutes les autres représentations possibles de l'ensemble considéré. Par contre, pour ce qui concerne le critère du nombre de slice *registre*, la figure 3.17 montre que la précision (format B) dont la probabilité d'erreur de décision est de 9.8810^{-4} s'avère la plus intéressante. Nous pouvons dans ce cas conclure que la première précision, i.e. le format A, est un bon compromis entre le coût en consommation de puissance et les performances en termes de probabilité d'erreur de décision.

3.5 Conclusion

Nous avons traité dans ce chapitre le problème de propagation des erreurs de décision causées par l'arithmétique virgule fixe dans l'itération d'opérateur de décision au sein d'une structure récursive. Ce problème se rencontre dans de nombreuses applications de traitement du signal et de communications numériques, et plus particulièrement, dans l'opération d'égalisation à retour de décision. Pour cette application, nous avons présenté deux approches différentes pour estimer la probabilité d'erreur de décision dans le régime de convergence de l'égaliseur DFE. Les modèles analytiques proposés sont valides pour tout système constitué d'itération d'opérateurs de décision. Le premier modèle fournit une expression quasi-exacte de la probabilité d'erreur de décision et

il est basé sur la résolution d'un système non linéaire par la méthode de Newton Raphson. Cependant, ce premier modèle peut être limité par les grandes capacités de calcul requises dans le cas de constellations à haut rendement spectral (i.e. à grand nombre d'états). Le second modèle proposé est fondé quant à lui sur l'estimation d'une borne supérieure de la probabilité d'erreur de décision. Ce modèle a été utilisé pour choisir le format optimal en arithmétique virgule fixe pour réaliser une implémentation matérielle rigoureuse en termes de consommation de ressources et d'énergie. D'autres applications à structure itérative non linéaire telles que les turbo décodeurs et les décodeurs LDPC n'ont pas été évaluées et dimensionnées en format virgule fixe dans la littérature. C'est pour cette raison que nous consacrons le prochain chapitre à l'étude spécifique de ces applications.

Chapitre 4

Évaluation de la précision du turbo décodage

Sommaire

Les systèmes de communications modernes comptent sur l'opération de codage canal pour améliorer la fiabilité de la liaison de communication et par conséquent la qualité de service (QoS) à l'utilisateur final. Pour atteindre cet objectif, un bloc d'informations issues de la source est encodé en un mot de code comportant de la redondance de l'information de source sous forme de bits de parité. A la réception, ces bits de parité sont exploités par le décodeur pour effectuer la correction d'erreur en avant (*Forward Error Correction* FEC), signifiant la correction partielle ou complète des erreurs introduites par la transmission à travers un canal bruité.

Deux catégories principales de codes de canal ont gagné l'élan de la communauté scientifique et industrielle : les codes de contrôle de parité à faible densité [46] et la concaténation en série ou en parallèle des codes convolutifs [80] : on parle alors de SCCC (pour *Serial Concatenated Convolutional Code*) ou de PCCC (pour *Parallel Concatenated Convolutional Code*). Les codes LDPC ont été conçus en premier lieu par R. Gallager dans les années 60, ils ont été vite abandonnés à cause de la technologie micro électronique inadéquate et incapable de satisfaire à la complexité de l'algorithme de décodage. Dans le début des années 90, les codes de canal sont devenus plus populaires, lorsque C. Berrou et A. Glavieux ont inventé le turbo décodage des PCCCs [22], vite étendu aux SCCCs [80][81]. Ces travaux ont marqué l'entrée dans une nouvelle ère des communications numériques puisqu'ils ont ouvert la voie à de nombreuses activités de recherche et d'avancées dans le domaine de la théorie de l'information. Par ailleurs, l'évolution continue de la technologie VLSI (*Very Large Scale of Integration*) a renforcé le succès des turbo codes et codes LDPC. Les processus CMOS submicroniques permettent l'implémentation des décodeurs en maintenant une fréquence d'horloge très élevée et donc ceci permet d'avoir un très

haut débit. Actuellement, divers standards de communication spécifient l'utilisation de turbo ou LDPC codes ou les deux pour la correction d'erreurs. Ceci couvre différentes applications et services tels que les réseaux d'accès i.e. les réseaux d'accès locaux sans fil (W-LANs) [9] et les réseaux d'accès métropolitains sans fil tels que la norme WIMAX [8], les réseaux cellulaires à haute vitesse avec tout d'abord la norme UMTS-2000 [1], puis la norme 3GPP [6], jusqu'à l'évolution vers la 3GPP-LTE [7], la communication par satellite [2] [4], les terminaux portatifs [5] et les liens de données à très haut débit sur fibre optique [3]. Chaque standard spécifie les paramètres de ses codes tels que la longueur des blocs, le rendement du code et le débit de décodage. Par conséquent, la conception des décodeurs multi-standard représente un défi technologique essentiel afin de concevoir des architectures flexibles satisfaisant aux contraintes liées à la consommation en puissance et à la surface des puces.

La définition de l'architecture VLSI en virgule fixe de l'algorithme de décodage qui est flexible, utilise le plus faible nombre de bits et possède des très bonnes performances de correction d'erreur. Par la suite, elle est un moyen efficace d'atteindre une meilleure implémentation du décodeur considéré, en tenant compte d'une complexité réduite et d'une faible consommation de puissance. D'autre part, les unités de traitement de signal numérique (DSP) en virgule flottante ou en virgule fixe sont inadéquates pour cet objectif. En plus de leur consommation de puissance importante, ces unités DSP ne répondent qu'aux exigences de débit des normes les plus lentes, et seulement avec des hauts degrés de parallélisme et donc avec une consommation de puissance en hausse.

Pour cette raison, nous développons ci-après un modèle précis en virgule fixe pour un décodeur turbo et un décodeur LDPC, et ceci dans un cadre unifié exploitant les analogies communes à ces deux types de codes et leurs algorithmes de décodage associés. Dans la première partie, nous décrivons la problématique de l'optimisation de la largeur des données en virgule fixe. Nous présentons ensuite le principe du turbo décodage par le biais de l'algorithme BCJR (Bahl, Cocke, Jelinek and Raviv) appliqué aux codes convolutionnels. Dans une seconde étape, nous proposerons un modèle en virgule fixe pour le turbo décodeur en détaillant les choix relatifs à la dynamique et à la quantification des opérations au sein du décodeur. Finalement, nous présenterons les performances obtenues par le décodeur LDPC proposé en analysant les courbes de taux d'erreur par paquet ou FER (pour *Frame Error Rate*).

4.1 L'optimisation de la largeur des données en virgule fixe

Le coût et la précision des systèmes implémentés en utilisant l'arithmétique virgule fixe sont fonction des longueurs de mot de code assigné. Étant donné qu'il existe plusieurs possibilités pour sélectionner la largeur d'un mot de code, le choix du format virgule fixe pour chaque signal doit être effectué avec précaution. Pour un algorithme donné, nous définissons w le vecteur constitué par les longueurs de mot de code de chaque entrée dans l'implémentation en virgule fixe. Nous pouvons alors définir l'expression $C(w)$ de la fonction d'évaluation en virgule fixe ainsi que la métrique de précision $\lambda(w)$ en fonction de ce seul paramètre w.

4.1.1 Variantes du problème

Afin d'optimiser le processus de conversion d'un algorithme en arithmétique fixe, il est nécessaire de réaliser un compromis en coût d'implémentation et précision des calculs. Ce compromis peut être établi de manière (i) à obtenir les meilleures performances en imposant une contrainte sur les coûts en consommation (puissance, ressources), ou bien alors de manière (ii) à limiter la complexité d'implémentation pour une contrainte en performance minimale. Par conséquent, le problème d'optimisation de la spécification en virgule fixe d'un algorithme peut être mené suivant deux variantes principales. Nous discutons ci-dessous des principales caractéristiques de ces deux approches.

Problème de maximisation de performance

Dans ce problème, la contrainte principale concerne le coût d'implémentation. Étant donné le budget du coût maximal C_{budget}, l'objectif consiste à maximiser la précision du système λ. Ce problème peut être formulé comme suit :

$$\max(\lambda(w)) \qquad sous \quad contrainte \qquad C(w) \leq C_{budget}. \tag{4.1}$$

Problème de minimisation du coût

Dans ce problème, les performances de l'algorithme spécifié en virgule fixe imposent une contrainte sur le problème d'optimisation. Étant donné le bruit de quantification maximal $\lambda_{objectif}$, l'objectif du problème d'optimisation est de minimiser le coût C de son implémentation. Ceci s'écrit formellement par :

$$min(C(w)) \qquad sous \quad contrainte \qquad \lambda(w) \leq \lambda_{objectif}. \tag{4.2}$$

La sélection entre les deux versions du problème d'optimisation est dictée par les exigences de la conception. Le concepteur peut choisir de résoudre le problème de minimisation de coût s'il a des contraintes sévères de précision en laissant une marge suffisante pour le coût total. Par ailleurs, le choix du concepteur peut être la maximisation de performance s'il y a des contraintes sévères sur le coût d'implémentation tout en faisant le compromis sur la précision des calculs. Dans une situation idéale, le compromis établi en résolvant les problèmes pour diverses contraintes cibles est unique et identique. Ceci correspond à la solution optimale de Pareto [75] d'un système donné. Malheureusement, les solutions à ces problèmes d'optimisation sont souvent obtenues à partir d'algorithmes heuristiques ce qui peut parfois conduire à une solution qui ne corresponde pas l'optimal absolu.

4.1.2 Variable d'optimisation

Un système de traitement de signal correspond à des centaines d'opérations en virgule fixe. A titre d'exemple, l'algorithme de FFT à 64 points possède 960 opérations incluant des additions et des multiplications avec des constantes. La partie entière et la partie fractionnaire des formats en virgule fixe sont choisies de façon à fournir un niveau de dynamique et une précision suffisante pour tous les signaux. Il est possible d'avoir des longueurs de données flexibles pour les architectures flexibles comme un FPGA ou les architectures des DSP modernes avec un support SIMD. Afin de résoudre le problème d'optimisation des longueurs des données, chaque choix de la longueur d'un mot de code pour chaque opération en virgule fixe doit être optimal.

Soit M le nombre des signaux dans le système. Si chaque signal possède N configurations possibles pour la longueur de son mot de code, il existe N^M combinaisons possibles de vecteurs de longueur de mot de code. Ceci exprime le paradigme de l'affectation multiple de longueurs de mot de code pour l'amélioration en virgule fixe.

Affecter la même longueur du format pour chaque signal réduit la complexité du problème d'optimisation. Ceci est connu comme le paradigme d'affection uniforme des longueurs de mot de code pour le raffinement virgule fixe. Dans cette approche, toutes les opérations sont effectuées par la même longueur. En d'autres termes, tous les signaux et les opérateurs dans le système possèdent le même format en virgule fixe. Cette approche pour la conception en virgule fixe des systèmes de traitement de signal est utilisée pour l'implémentation des architectures de chemin de données de largeur fixe. Le format en arithmétique virgule fixe est choisi de telle sorte que la partie entière est la partie fractionnaire soient suffisamment larges pour que la contrainte de précision soit vérifiée pour tous les signaux. Si le paradigme uniforme de la longueur de mot de code devait être utilisé, le nombre de combinaisons de longueurs des mots de code qui sont nécessaires avant d'atteindre l'optimalité est N, où N est le nombre de configurations de longueur de mot de code réalisables dans l'architecture donnée.

4.1.3 Solution au problème d'optimisation de la longueur du mot de code

Auparavant, le problème de raffinement en virgule fixe était traité en utilisant le paradigme uniforme de longueur de mot de code (ULM). Dans le paradigme ULM, tous les signaux sont assignés à la même longueur de mot de code de virgule fixe et au même format. Par contre, chaque signal est assigné à un format virgule fixe approprié en fonction de sa sensibilité à la sortie du système. Il a été montré que le paradigme de multiple longueurs de mot de code (MLM) donne parfois de meilleurs résultats que les formats virgule fixe obtenus en utilisant le paradigme ULM [42]. Les avantages de la résolution du problème de raffinement de virgule fixe en utilisant le paradigme MLM se fait au prix de la résolution du problème d'optimisation NP-hard. Bien que les solutions de raffinement virgule fixe soient simplement $O(N)$, où N est le choix varié du format de longueur du mot de code si le diagramme ULM est utilisé à la place.

Des techniques variées pour résoudre les problèmes d'optimisation de la longueur du mot de code peuvent être catégorisées en approches à base optimale, stochastique et heuristique-rapide [75].

4.2 Optimisation du turbo décodage

Nous présentons dans cette section un modèle en virgule fixe du décodeur turbo afin d'optimiser les formats en virgule fixe des différentes quantités impliquées dans l'algorithme de décodage et ceci dans le but d'améliorer les performances en terme de taux d'erreur par paquet FER. En premier lieu, nous décrivons dans le paragraphe ci-après le principe général du décodage turbo, puis nous présentons le modèle proposé en virgule fixe en se basant sur le fonctionnement de l'algorithme de turbo décodage. Enfin, dans la dernière partie, nous analysons les performances de l'algorithme de turbo décodage en fonction des différents choix possibles des formats de données en virgule fixe.

4.2.1 Présentation de l'algorithme

Avant de présenter l'algorithme de décodage turbo, il est fondamental de décrire le principe des codes turbo pour bien expliciter le fonctionnement de l'algorithme de décodage

Les codes turbo

En se concentrant sur la classe des codes convolutionnels concaténés parallèles PCCC (*Parallel Concatenated Convolutional Codes*), la figure 4.1 montre le codeur du turbo code pour la norme 3GPP-LTE. Ce dernier est composé de deux codeurs systématiques, récursifs, en parallèle où l'unité supérieure et inférieure sont alimentées par une version directe et une version entrelacée des bits d'information respectivement. L'entrelacement des bits du mot de l'information est effectuée dans le bloc Π de la figure 4.1. Chaque codeur RSC (*Recursive Systematic Code*) est un registre à décalage à retour linéaire LFSR (*Linear-Feedback Shift-Register*) particulier dont les bits de sortie c_i, $i = 0, 1$, appelés aussi bits de parité, sont déterminés en fonction de l'état S du registre, des connections en avant et en arrière, mais également de la valeur du bit u présent à l'entrée du codeur.

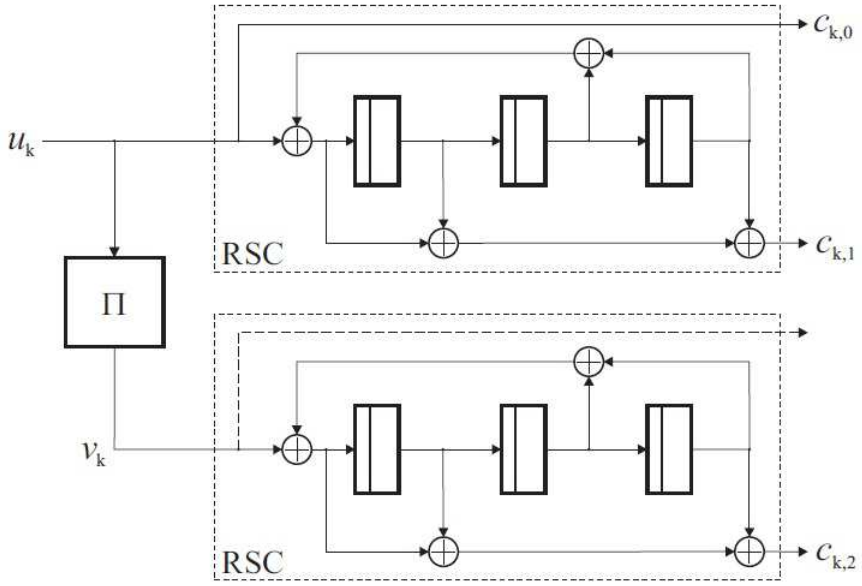

FIGURE 4.1 – Turbo codeur de la norme 3GPP-LTE

Les performances du turbo code dépendent des paramètres des RSC tels que le nombre d'états, noté v, et le lien de retour. Le nombre d'états v est lié au nombre d'éléments de mémoire dans le RSC qui est lui-même relié à la longueur de contrainte L du code ($L = 4$ dans l'exemple de la figure 4.1) suivant la relation $v = 2^{L-1}$.

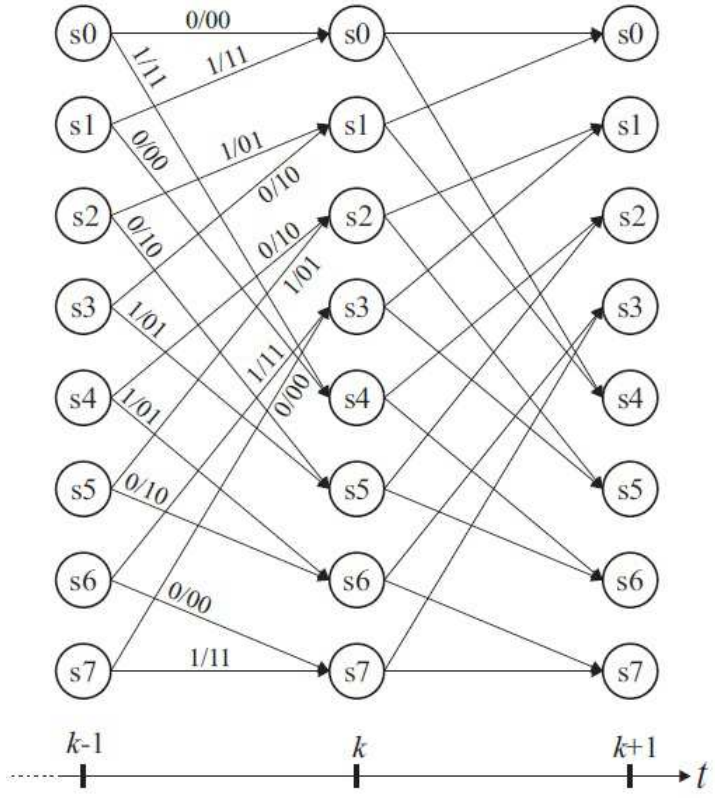

FIGURE 4.2 – Exemple d'un treillis à 8 états

Le processus de codage du RSC peut être effectivement représenté en ayant recours au graphe de treillis représenté par la figure 4.2 pour un codeur 3GPPP-LTE. Il s'agit d'un diagramme montrant l'évolution en temps de l'état LFSR et décrivant les transitions entre les paires des états consécutifs comme indiqué sur la figure 4.2.

Chaque branche du treillis reliant deux états consécutifs est labellisée par la paire de symboles d'information présents en entrée des RSC et par les bits de parité produits en sortie du codeur turbo. Ainsi, pour une suite de bits d'information donnée, le processus de codage peut être décrit par un chemin spécifique au sein des différentes branches décrivant le treillis du code.

Visant à améliorer les capacités de correction d'erreur, les turbo codes M-aires sont devenus largement utilisés dans les standards de communication récents après leur introduction dans les années 2000 [23]. Dans ce cas, chaque symbole d'information prend ses valeurs dans un ensemble comportant $M > 2$ valeurs d'amplitude possibles ($M = 2$ correspond à un code binaire). Chaque symbole peut être exprimé sur m bits, donc $M = 2^m$. Des valeurs importantes de M peuvent être envisagées afin d'améliorer le pouvoir de correction d'erreur du turbo décodeur, mais ces valeurs ne sont pas d'une grande utilisation en pratique en raison de la complexité excessive de l'algorithme de décodage associé.

L'algorithme de décodage BCJR (Bahl-Cocke-Jelinek-Raviv)

L'algorithme BCJR est un passage obligé pour le turbo décodage puisqu'il est appliqué au décodage des deux composantes de décodage RSC du turbo code [63]. La figure 4.3 résume les notations utilisées par l'algorithme de décodage BCJR d'un code convolutionnel M-aire ($M = 2^m$).

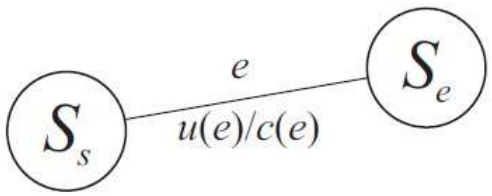

FIGURE 4.3 – Les notations de BCJR dans le treillis

La branche orientée e connecte l'état de départ $S_S(e)$ à l'état d'arrivé $S_e(e)$. $u(e)$ est l'information symbole liée à la branche e, issue de l'ensemble $U = \{0, 1, ..., M-1\}$, avec $M = 2^m$. Pour un symbole d'information $u(e)$ présent en entrée du codeur et une mémoire correspondant à l'état S_s, le codeur génère à sa sortie un symbole codé $c(e)$ associé à la branche e dont les éléments sont les bits codés $c_i(e)$, $i = 0, 1, ..., n-1$ de $c(e)$. Alors, pour m bits d'information codés dans le symbole u, $n \geq m$ bits codés sont générés et le rapport $r = m/n$ est appelé le rendement du code.

Correspondant à une forme particulière du décodage MAP, l'algorithme BCJR vise à la maximisation de la probabilité *a posteriori* du bit transmis pour une observation donnée du mot de code reçu dans le bruit. Pour une implémentation efficace, l'algorithme est formulé en termes de messages de fiabilités sous la forme des rapports logarithmiques de vraisemblance LLRs (Log-Likelihood Ratios). Soit x une variable aléatoire $M - aire$ dont la valeur est dans l'ensemble $X = \{x_0, x_1, ..., x_{M-1}\}$. Son LLR est défini par :

$$LLR(x = x_i) = \log \frac{P(x = x_i)}{P(x = x_0)} \tag{4.3}$$

avec $i = 1, 2, ..., M-1$. x_0 est utilisé comme le symbole de référence pour la normalisation, par conséquent seulement $M-1$ LLRs sont associés pour une variable aléatoire $M - aire$.

L'algorithme BCJR implique les quantités suivantes dans son fonctionnement de décodage (nous utilisons les mêmes notations que [81]) :

- $\lambda_{k,i}^{ch}$ est l'information canal *a priori* pour le bit codé c_i à l'instant k, avec $i = 0, 1, ..., n-1$ et $k = 0, 1, ..., N-1$. $\lambda_{k,i}^{ch}$ fréquemment nommé entrée LLR, désigne l'entrée de l'algorithme BCJR.
- $\gamma_k(c(e))$ (ou simplement $\gamma_k(e)$) est la métrique cumulative associée au symbole codé $c(e)$ sur la branche e à l'instant k. $\gamma_k(c(e))$ est fréquemment appelée *métrique de branche*.
- $\lambda_k^I(u(e))$ (ou simplement $\lambda_k^I(e)$) est l'information *a priori* associée au symbole d'information $u(e)$ sur la branche e à l'instant k.
- $\lambda_k^O(u(e))$ (ou simplement $\lambda_k^O(e)$) est l'information extrinsèque *a posteriori* associée au symbole d'information $u(e)$ sur la branche e à l'instant k.
- $A_k^{APP}(u(e))$ (ou simplement $A_k^{APP}(e)$) est la probabilité *a posteriori* APP associée au symbole d'information $u(e)$ sur la branche e à l'instant k.

L'algorithme BCJR commence par calculer les métriques branches $\gamma_k(e)$ par la relation suivante :

$$\gamma_k(e) = \sum_{i=0}^{n-1} c_i(e).\lambda_{k,i}^{ch} \tag{4.4}$$

avec $k = 0, 1, ...N-1$ est l'index du treillis.

La métrique branche $\gamma_k(e)$ et l'information extrinsèque *a priori* $\lambda_k^I(e)$ conduisent à calculer les récursions *forward* et *backward* $\alpha^{'}$ et $\beta^{'}$, calculées dans le domaine logarithmique par les relations suivantes :

$$\alpha_{k+1}^{'}(S_i) = \overset{*}{\max}_{e:S_E(e)=S_i} \{\alpha_k^{'}(S_S(e)) + \gamma_k(e) + \lambda_k^I(e)\} \tag{4.5}$$

$$\beta_k^{'}(S_i) = \overset{*}{\max}_{e:S_S(e)=S_i} \{\beta_{k+1}^{'}(S_E(e)) + \gamma_k(e) + \lambda_k^I(e)\} \tag{4.6}$$

avec $\overset{*}{\max}(a, b)$ est l'opérateur défini par :

$$\overset{*}{\max}(a, b) = max(a, b) + \log(1 + e^{-|a-b|}). \tag{4.7}$$

Cependant, l'opérateur max^* peut être approximé par une simple opération max pour les implémentations à faible complexité. Les récursions *forward* et *backward* α et β sont évaluées à travers l'ensemble des branches e entrantes et sortantes respectivement d'un état S_i à l'instant $k+1$ pour la métrique *forward* α, et à l'instant k pour la métrique *backward* β. Ces récursions sont initialisées avec $\alpha_0^{'} = \alpha_{init}$ et $\beta_N^{'} = \beta_{init}$ à l'instant $k = 0$ et $k = N$, respectivement.

Fermeture du treillis : transformation en code en bloc Tout code convolutif est convertible en un code en bloc. Il suffit de stopper l'avance dans le treillis après un nombre de branches fixé par la longueur du code en blocs recherché. Une méthode triviale consiste à couper le treillis sans spécifier l'état final (troncature). La bonne méthode transforme le code convolutif en un code en bloc en forçant le retour à l'état 0 par une terminaison du treillis.

En effet, les valeurs d'initialisation dépendent de la stratégie de terminaison sélectionnée. Ainsi, (i) pour des codes sans terminaison, les valeurs d'initialisation sont [1/v,...,1/v] ; (ii) pour les codes dont la terminaison est zéro, ces valeurs sont égales à $[1, 0, ..., 0]$ et, (iii) pour des codes circulaires, les valeurs d'initialisation des métriques de chemins (*forward* et *backward*) correspondent aux valeurs obtenues à l'itération précédente. Les récursions métriques sont sous forme des probabilités logarithmiques. Afin d'augmenter la robustesse de l'algorithme [72][37], elles sont normalisées. La valeur prise comme référence est typiquement celle de l'état S_0, ce qui conduit aux récursions normalisées suivantes :

$$\alpha_k(S_i) = \alpha_k^{'}(S_i) - \alpha_k^{'}(S_0) \tag{4.8}$$

$$\beta_k(S_i) = \beta_k^{'}(S_i) - \beta_k^{'}(S_0). \tag{4.9}$$

Une fois que les récursions *forward* et *backward* sont disponibles pour tous les états du treillis, l'estimation *a posteriori* du symbole d'information u est déduite par :

$$\lambda_k^O(u_i) = \underset{e:u(e)=u_i}{\overset{*}{\max}} \{\alpha_k(S_S(e)) + \gamma_k(e) + \beta_{k+1}(S_E(e))\} - \underset{e:u(e)=u_0}{\overset{*}{\max}} \{\alpha_k(S_S(e)) + \gamma_k(e) + \beta_{k+1}(S_E(e))\} \tag{4.10}$$

Comme elle n'est pas directement liée au message *a priori* en entrée $\lambda_k^I(e)$, la probabilité APP en sortie $\lambda_k^O(u_i)$ est dite extrinsèque.

Le principe du turbo décodage

L'algorithme de turbo-décodage est obtenu par application directe de l'algorithme BCJR à ces deux composantes RSC selon le diagramme de la figure 4.4 où les deux unités marquées SISO-1 et SISO-2 sont des unités SISO (*Soft-in Soft-out*). L'algorithme évolue au cours du temps par l'échange itératif des messages extrinsèques qui sont les sorties *a posteriori* des blocs SISO. L'algorithme est alimenté par les estimations canal *a priori* $\lambda_{k,i}^{ch}$, sous la forme de LLR et calculées selon l'équation (4.3) pour des symboles binaires ($M = 2$). La sortie du bloc SISO-1, appelée $\lambda^{ext,1}$ dans la figure 4.4, est entrelacée par l'entrelaceur Π avant d'être transmise au bloc SISO-2 comme une information *a priori*. Ce dernier bloc reçoit également en parallèle une version entrelacée de l'estimation du canal a priori $\lambda_{k,i}^{ch}$, puis il fournit en sortie un message de fiabilité *a posteriori* $\lambda^{ext,2}$. Après inversion de l'opération d'entrelacement, cette fiabilité est adressée à l'entrée du bloc SISO-1 en tant qu'estimation *a priori* raffinée des symboles transmis.

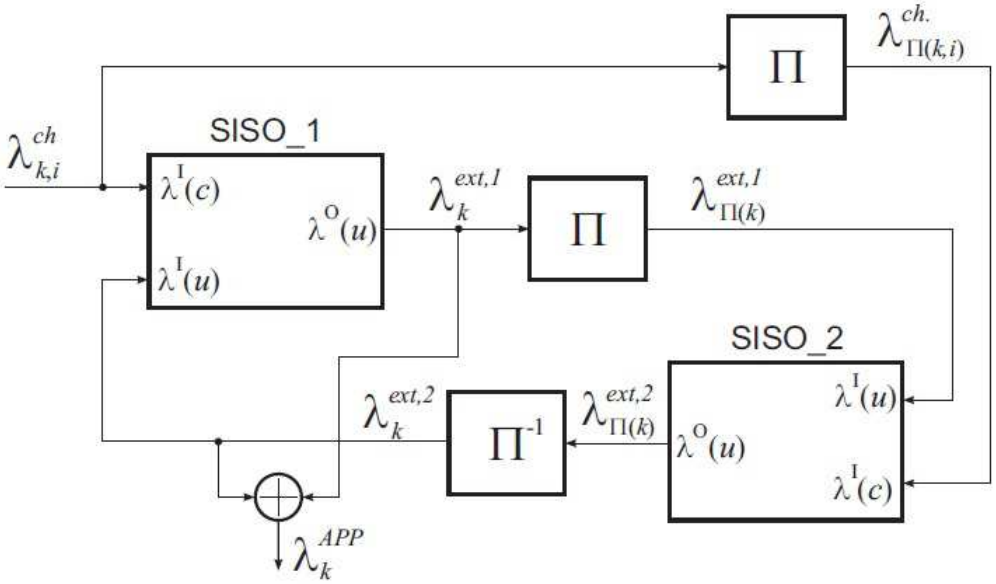

FIGURE 4.4 – Le principe du turbo décodage

Comme montré dans la figure 4.4, la sortie du turbo décodeur est donnée par l'estimation *a posteriori* A_k^{APP} du symbole transmis. Elle est obtenue par la sommation des deux messages extrinsèques (non entrelacées) disponibles en sortie des deux unités SISO. Elle est donnée par :

$$A_k^{APP}(u_i) = \lambda_k^{ext,1}(u_i) + \lambda_k^{ext,2}(u_i) \tag{4.11}$$

avec $u_i \in U = \{u_0, u_1, ..., u_{M-1}\}$ et $k = 0, 1, ..., K-1$.

4.2.2 Modélisation et *design* en virgule fixe

Considérons un système de numération de base B, la représentation en virgule fixe X d'un signal réel x est exprimée par :

$$X = \sum_{n=0}^{N_I-1} a_n B^n + \sum_{n=1}^{N_F} b_n B^{-n} \tag{4.12}$$

avec N_I et N_F sont respectivement le nombre d'entier appartenant à l'ensemble $D = \{0, 1, ..., B-1\}$ et représentant la partie entière et fractionnaire de x. Le nombre $N_x = N_I + N_F$ est le nombre total de nombres utilisés pour représenter x en base B.

La multiplication de (4.12) par le facteur B^{N_F} appelé facteur d'échelle est pratique pour se débarrasser du point décimal et elle est efficace pour une implémentation DSP en virgule fixe. Dans le cas d'un système binaire avec $B = 2$, X devient un nombre entier sous la forme de :

$$X = \sum_{n=0}^{N_x-1} x_n 2^n \tag{4.13}$$

où $x_n, n = 0, 1, ..., N_x - 1$ sont les chiffres binaires de la représentation entière de x, avec $x_n = B_{N_F-n}$ pour $n = 0, 1, ..., N_F - 1$ et $x_n = a_{n-N_F}$ pour $n = N_F, N_F + 1, ..., N_F + N_I - 1$.

Loi de conversion

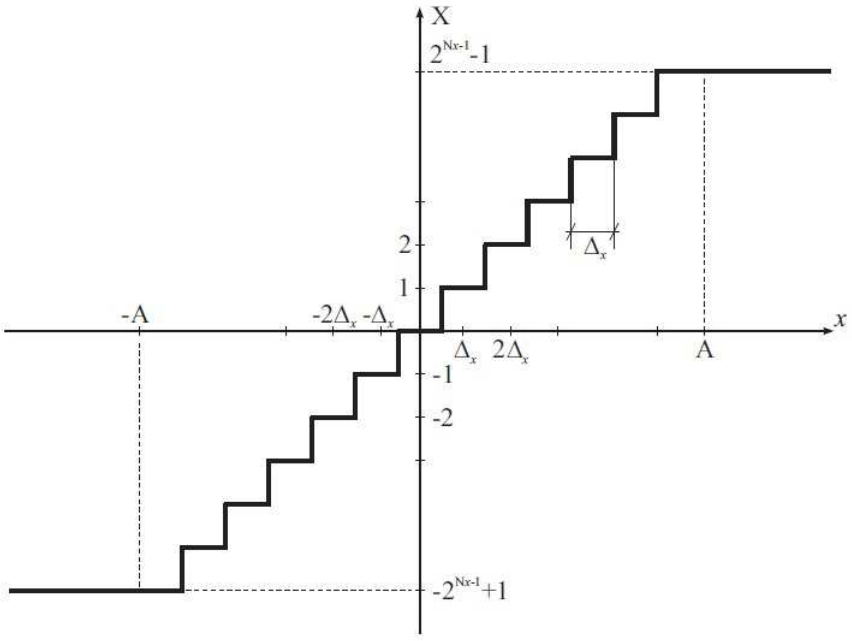

FIGURE 4.5 – Fonction en escalier de conversion de la virgule flottante à la virgule fixe

Soit un signal x défini dans le domaine des réels, $x \in R$, sa représentation virgule fixe X sur N_x bits est donnée selon une loi bien précise. Comme seulement un nombre limité N_x de bits est disponible, le domaine de x doit être contraint à un intervalle de R, soit $[-A, A]$. Donc, une saturation préventive du signal dans l'intervalle $[-A, A]$ doit être effectuée et la valeur de A sera interprétée comme une dynamique. L'opération de la conversion en virgule fixe peut être effectuée suivant la transformation suivante :

$$\begin{cases} X = \min(2^{N_x-1}, \lfloor \frac{x}{\Delta_x} + 0.5 \rfloor), x \geq 0 \\ X = \max(2^{N_x+1}, \lfloor \frac{x}{\Delta_x} - 0.5 \rfloor), x < 0 \end{cases} \tag{4.14}$$

où $\Delta_x = \frac{2A}{2^N-1}$ est le pas de quantification, c'est à dire la différence maximale entre deux différentes valeurs flottantes qui sont codées dans le même symbole X en virgule fixe. La valeur de Δ_x est une mesure de la résolution de la représentation. En d'autres termes, c'est le poids du bit le moins significatif (*Least Significant Bit* LSB) x_0 de X.

Il convient de préciser que (4.14) effectue non seulement la quantification du signal d'entrée, mais elle limite son domaine à l'intervalle $[-A, A]$, comme le montre la figure 4.5. Les valeurs supérieures (inférieures) à $A(-A)$ sont saturées au plus grand niveau positif $2^{N_x-1} - 1$(le plus petit niveau négatif $-2^{N_x-1} - 1$). Dans (4.14), seulement 2^{N_x-1} niveaux de virgule fixe sont utilisés (le co-domaine de la fonction de transformation est symétrique par rapport au niveau 0), ce choix empêche l'algorithme de dériver vers des valeurs négatives comme montré dans [37].

Ainsi, la paire (A, N_x) définit complètement la quantification du signal en virgule flottante x, et fournissant la dynamique et le poids de LSB Δ_x utilisé pour sa représentation. Cette approche est similaire a celle décrite dans [37] et [30] pour la quantification des entrées LLRs, et elle est plus flexible que celle généralement adaptée dans la littérature [72][19][78], où le format virgule fixe est spécifié par les paires (N_I, N_F), sans tenir compte de la dynamique du signal réel sous-jacent.

En d'autres termes, la dynamique du signal réel est souvent mise sous la forme $A = 2^{N_I-1}$ et elle est limitée à une puissance de deux. Par contre, la représentation que nous proposons dans cette thèse dépasse cette restriction. Elle sera appliquée à chaque élaboration interne en virgule fixe du turbo décodeur.

Modélisation virgule fixe du turbo décodeur

Plusieurs travaux dans la littérature ont traité cette problématique de modélisation en virgule fixe des algorithmes de décodage itératif [72][39][30]. De plus, des indications utiles sont fournies pour l'implémentation pratique du turbo décodeur [19][12] et du décodeur LDPC [88][78]. En outre, les modèles dans [72] et [37] fournissent des bornes pour l'augmentation maximale des signaux impliqués à l'intérieur du décodeur Turbo.

L'algorithme de turbo décodage est reformulé dans le domaine de l'arithmétique virgule fixe pour impliquer les opérations avec des entiers. Suivant une approche en cascade, toutes les opérations impliquées dans l'algorithme sont successivement converties en leur représentant en virgule fixe une après l'autre.

1-L'information canal *a priori* : Les LLRs canal sont quantifiés selon (4.14) en utilisant un seuil $A_{\lambda^{ch}}$ et $N_{\lambda^{ch}}$ bits.

2-La métrique de branche : Le calcul de $\gamma_k(e)$ dans (4.4) implique la somme de messages de fiabilités canal *a priori* $\lambda_{k,i}^{ch}, i = 0, 1, ..., n-1$. Par conséquent, dans le pire des cas, la somme cohérente $\gamma_k(e)$ est donnée par :

$$\gamma_k(e) = n.A_{\lambda^{ch}} \tag{4.15}$$

La représentation en virgule fixe Γ de γ doit être représentée avec :

$$A_\gamma = n.A_{\lambda^{ch}} \tag{4.16}$$

et le nombre de bits de γ est donné par :

$$N_\gamma = N_{\lambda^{ch}} + \lceil \log_2(n) \rceil. \tag{4.17}$$

3-L'opérateur max^* : L'opération $z = max^*(x, y)$ implique le calcul du max des deux signaux x et y, et l'addition du terme correctif dans l'intervalle $]0, log2]$. Par conséquent, la dynamique de z est majorée par A_z. Elle est donnée par la relation suivante :

$$A_z = max(A_x, A_y) + log2. \tag{4.18}$$

Afin de laisser la comparaison possible, les versions en virgule fixe X et Y de x et y respectivement doivent avoir la même résolution, soit donc $\Delta_x = \Delta_y = \Delta$. Sous cette hypothèse, le nombre de bits nécessaires pour représenter z peut être déduit de la définition de Δ comme suit :

$$2^{N_z} = \frac{2.A_z}{\Delta} + 1 = \frac{2.A}{\Delta} + \frac{2.log2}{\Delta} + 1 = (2^N - 1) + \frac{2.log2}{\Delta} + 1 = 2^N(1 + \frac{log2}{A}) \tag{4.19}$$

avec $A = max(A_x, A_y)$ et $N = max(N_x, N_y)$. Par conséquent, nous supposons que $A > log2$ ce qui est généralement le cas, alors l'expression (4.19) donne :

$$N_z = \lceil \log_2 \left(2^N \left(1 + \frac{log2}{A} \right) \right) \rceil = N + 1. \tag{4.20}$$

Par contre, les équations (4.19) et (4.20) sont strictement maintenues lorsque $x = A_x = y = A_y = A$ i.e. lorsque la contribution du terme correctif est maximale. En plus de ce cas très défavorable, le bit additionnel exprimé dans (4.20) n'est pas réellement exploité et l'utilisation de $A_z = max(A_x, A_y)$ est généralement suffisante. Par conséquent, le résultat de l'opérateur $max*$ peut être saturé sur $N_z = N$ bits. Cette approximation conduit à de très faibles pertes d'information et par la suite possède un impact négligeable sur les performances de l'algorithme. Ainsi, l'opération max^* en virgule fixe devient :

$$Z = max^*(X, Y) = min\{max(X, Y) + LUT(D), L\} \tag{4.21}$$

avec $L = 2^{N-1} - 1$ est le seuil de saturation. Dans cette dernière expression, le terme correctif est quantifié en utilisant la même résolution Δ que les entrées x et y et il est enregistré dans une table de recherche LUT (*look-up table*) adressée par $D = |X - Y|$.

4-L'information extrinsèque *a posteriori* Puisque les messages de fiabilités extrinsèques *a posteriori* et les récursions *forward/backward* sont mutuellement dépendants à travers le principe turbo itératif, leur représentation en virgule fixe peut être étudiée sous l'hypothèse que les récursions métriques d'état soient représentées sur $N_\alpha = N_\beta = N_\gamma + h$ bits, où h est un entier. D'après l'équation (4.10), la dynamique de λ^O est majorée par :

$$A_{\lambda^O} = (A_\alpha + A_\beta + A_\gamma) - (-A_\alpha - A_\beta - A_\gamma) = 2.(2A_\alpha + A_\gamma) = 2A_\gamma.(1 + 2^{h+1}) \tag{4.22}$$

avec $A_\alpha = 2^h A_\gamma$ et $A_\alpha = A_\beta$.

Nous pouvons alors exprimer la représentation de la précision de λ^O à partir de la relation $N_{\lambda^O} = \lceil log_2(2A_{\lambda^O}/\Delta_{\lambda^O} + 1) \rceil$ bits, ce qui donne :

$$N_{\lambda^O} = 1 + N_\gamma + \lceil log_2(1 + 2^{h+1}) \rceil = 1 + N_\gamma + \lceil max_2^*(0, h + 1) \rceil \tag{4.23}$$

avec la fonction max_2^* correspondant à l'opérateur max^* en base deux, et défini par :

$$max_2^*(a, b) = max^*(a, b) + log_2(1 + 2^{-|a-b|}). \tag{4.24}$$

Deux cas peuvent être distingués :

a) $h \geq 0$: Il est facile de prouver que $\lceil max_2^*(0, h + 1) \rceil = h + 2$, par conséquent, $N_{\lambda^O} = N_\gamma + h + 3 = N_\alpha + 3$.

b) $h < 0$: $\lceil max_2^*(0, h + 1) \rceil = 1$, $N_{\lambda^O} = N_\gamma + 2 = N_\alpha + 2 - h$.

Dans les deux cas, N_{λ^O} est une fonction connue de N_α et h.

5-Les récursions métriques d'état A cause de leur calcul itératif, l'amplitude des récursions forward/backward continue à augmenter tout au long du treillis si les récursions ne sont pas contrôlées par des moyens de saturations. Sous les mêmes hypothèses ($N_\alpha = N_\gamma + h$), l'augmentation des métriques d'état après une étape de mise à jour est majorée par :

$$2(A_\alpha + A_\gamma + A_{\lambda^I}) = 2A_\alpha(2 + 2^{-h} + A_{\lambda^I}/A_\alpha) \tag{4.25}$$

où l'information *a priori* λ^I est en effet la sortie *a posteriori* du bloc SISO précédent et donc $A_{\lambda^I} = A_{\lambda^O}$. En remplaçant (4.22) dans (4.25), cette dernière devient :

$$2A_\alpha(1 + 2^{-h} + 2.h^{-1}.2^{1-h}) = 2(5 + 2^{-h} + 2^{1-h})A_\alpha. \tag{4.26}$$

Nous pouvons donc conclure que la dynamique de α augmente d'un facteur de $2(5+3.2^{-h})$ après chaque itération du turbo décodeur ce qui se traduit par l'ajout de $1+\lceil log_2(5+3.2^{-h})\rceil$ bits. De même que précédemment, deux cas de figure peuvent se présenter :

- $h \geq 0$: le terme $2(5 + 3.2^{-h})$ est borné par 5 et 8, ce qui engendre l'addition de 4 bits à chaque itération.
- $h < 0$: le terme $\lceil log_2(5 + 3.2^{-h})\rceil$ est évalué à $2 - h$ et, au total, $3 - h$ éléments binaires sont ajoutés à chaque étape.

Par conséquent, la saturation de 4 ou $3 - h$ bits respectivement, empêche l'augmentation incontrôlée des métriques d'état représentées par (A_α, N_α). Dans [72] et [37], des bornes sont données pour la dynamique des récursions métriques d'état, et sont utilisées pour dimensionner la précision interne du bloc SISO. Par contre, dans l'approche que nous proposons, la résolution des métriques d'état est une entrée libre du modèle et elle est contrôlée par des moyens de saturation.

4.2.3 Réduction de la taille de mémoire

Le schéma complet du décodeur SISO est représenté à la figure 4.6. Une implémentation pratique d'un décodeur turbo code est basée sur l'utilisation étendue de mémoire comme un moyen pour échanger les messages extrinsèques et stocker les résultats intermédiaires qui correspondent aux métriques cumulées de chemin. Par conséquent, il est clair que la complexité totale du décodeur est fortement dominée par les mémoires.

Les techniques comme la troncature du bit de poids le plus faible LSB et la saturation du bit de poids le plus fort MSB sont très efficaces pour limiter la taille des données enregistrées dans la mémoire.

Par contre, l'utilisation de la saturation est préférable puisqu'elle réduit non seulement la taille de la mémoire, mais aussi les unités des chemins de données pour accéder à la mémoire. D'autre part, les données ayant subi des troncatures avant l'enregistrement, dans la mémoire, doivent être décalées à gauche après récupération de la mémoire pour restaurer la résolution originale car le poids du LSB Δ et les chemins de données ne bénéficient d'aucune réduction dans la taille.

Pour un signal x, les notation T_x et S_x désignent, respectivement, le nombre de LSB tronqués et MSB saturés avant le stockage dans la mémoire. Pour le turbo décodeur, la troncature et la saturation sont effectuées à chaque itération sur les métriques cumulées de chemins enregistrées dans la mémoire α/β-MEM (T_α et T_β respectivement) et sur l'information extrinsèque *a posteriori* stockée dans la mémoire A-MEM (T_A et S_A respectivement) comme indiqué sur la figure 4.6.

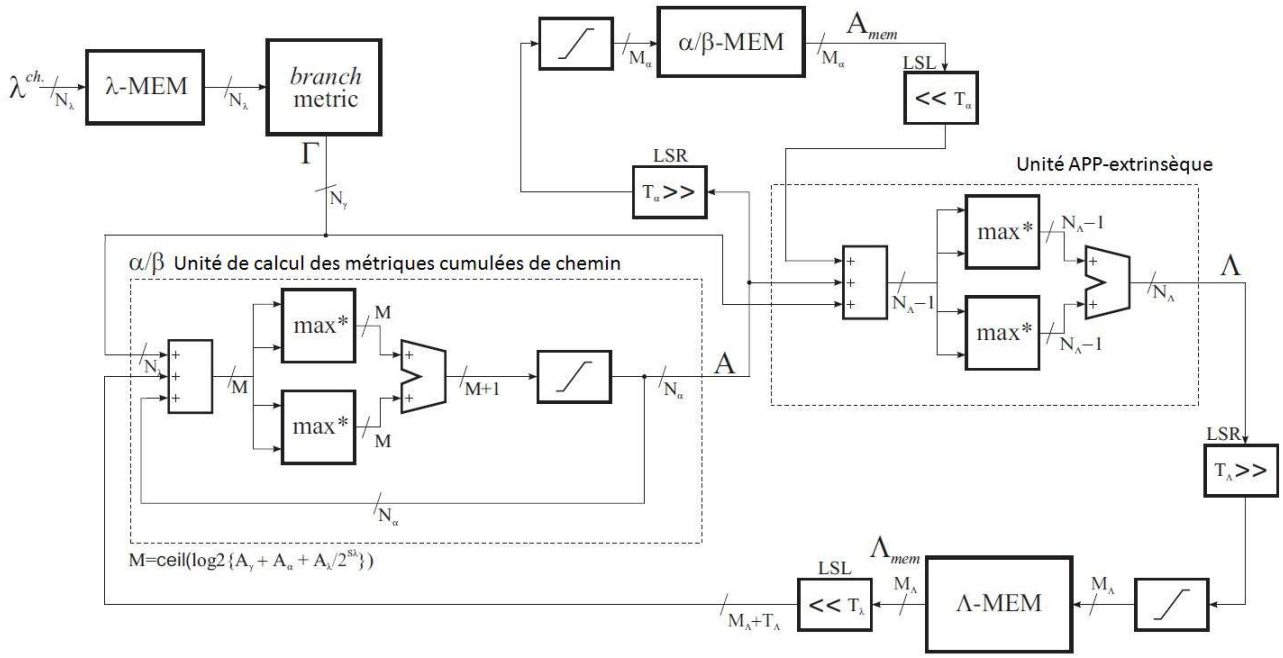

FIGURE 4.6 – Modèle virgule fixe du bloc SISO dans le turbo décodeur

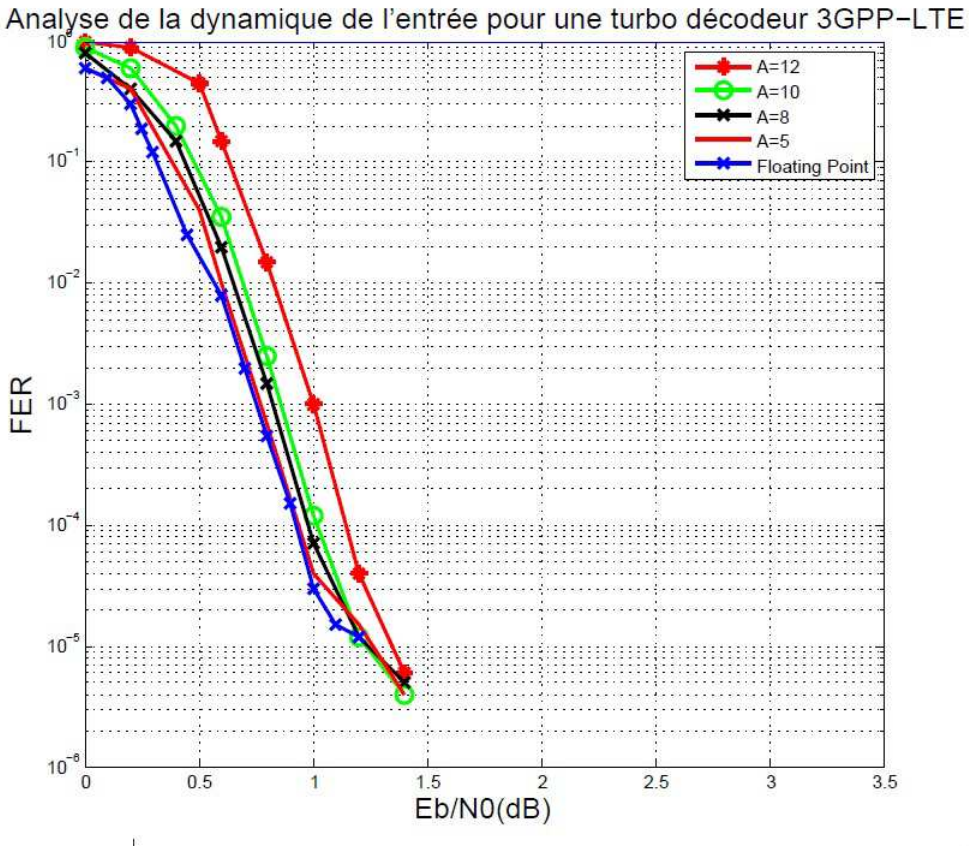

FIGURE 4.7 – Analyse de la dynamique pour $N_{\lambda^{ch}} = 5$ bits

4.2.4 Performance du turbo décodeur

Les performances de correction d'erreur et les pertes liées à l'implémentation du modèle virgule fixe développé dans la section précédente sont analysées par des simulations. Un système de communication complet formé du codeur, du modulateur, du canal de transmission introduisant un bruit additif blanc gaussien AWGN, du démodulateur et du décodeur décrit précédemment a été developpé en langage C avec une implémentation du modèle paramétrique en virgule fixe. Nous avons considéré un turbo décodeur satisfaisant à la norme 3GPP-LTE dont les paramètres sont décrits dans le tableau 4.1. Les performances en virgule fixe sont mesurées sous forme des courbes de taux d'erreur par paquet FER en fonction du rapport signal à bruit E_b/N_0 (SNR).

La norme	**Taille** m	**Longueur** K	**Taux** R	**Nombre d'itération** N_{it}
3GPP-LTE	1	1504	1/3	10

TABLE 4.1 – Les paramètres du code

Le premier point critique de la conception en virgule fixe concerne l'identification d'une valeur optimale de la dynamique de l'entrée LLR $A_{\lambda^{ch}}$. La figure 4.7 représente les performances FER pour différentes valeurs de $A_{\lambda^{ch}}$. Les entrées LLRs λ^{ch} sont codées sur $N_{\lambda^{ch}} = 5$ bits tandis que les métriques de chemins *forward* et *backward* sont représentées sur un nombre de bits suffisamment important ($N_\alpha = 6$) afin que les pertes d'implémentation ne soient causées que par la quantification des entrées λ^{ch}.

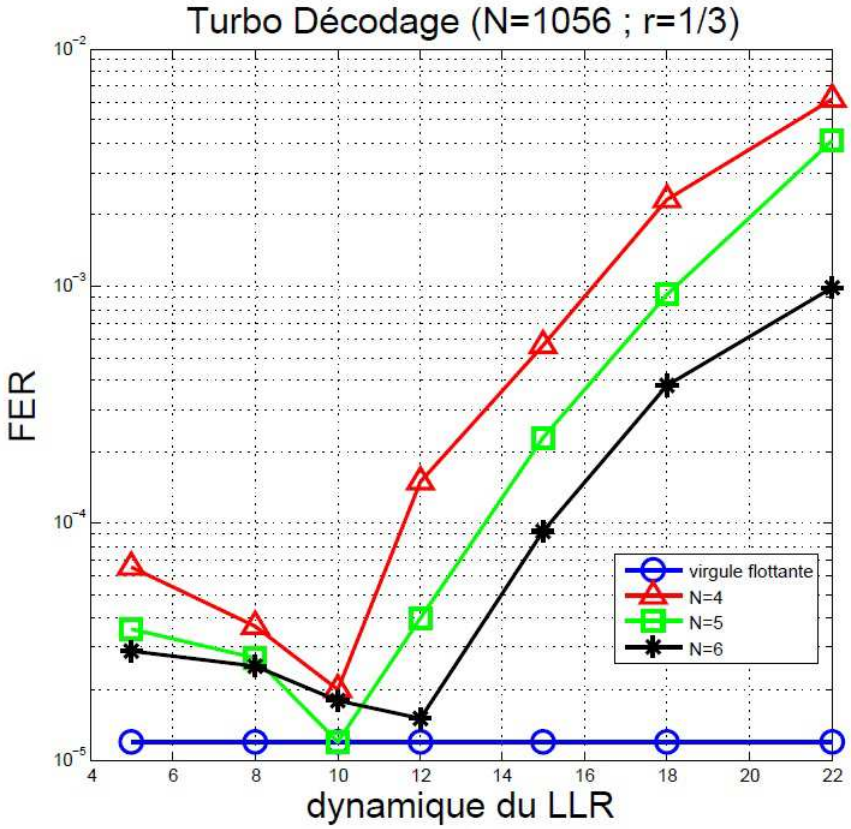

FIGURE 4.8 – Effet de la quantification des entrées LLRs

L'impact de la quantification des entrées LLRs est analysé par la figure 4.8 où la FER est tracée en fonction de la dynamique $A_{\lambda^{ch}}$ pour trois différentes valeurs de $N_{\lambda^{ch}}$ 4, 5 et 6 bits. Les trois courbes sont obtenues pour un $E_b/N_0 = 1.2dB$, avec une simulation en virgule flottante

comme référence. D'après cette figure, la valeur $A_{\lambda^{ch}} = 10$ semble être la meilleure pour un nombre de bits de 4 et 5. Tandis que $A_{\lambda^{ch}} = 12$ est le meilleur choix pour 6 bits.

D'après la figure 4.7, nous constatons que, plus que la valeur de $A_{\lambda^{ch}}$ est petite, plus les pertes dues à l'implémentation sont faibles. Le cas où $A_{\lambda^{ch}} = 10$ correspond à une détérioration majorée par $0.1dB$ par rapport au modèle de référence en virgule flottante, tandis que l'augmentation de la dynamique conduit à une résolution $\Delta_{\lambda^{ch}}$ très grossière, ce qui engendre des pertes considérables particulièrement pour les faibles SNR.

Si l'on s'intéresse maintenant aux performances obtenues pour une faible dynamique, i.e. pour $A_{\lambda^{ch}} = 5$, ce paramétrage correspond à une quantification grossière des entrées canal LLRs car de nombreux échantillons en virgule flottante vont être saturés à la valeur $\pm A_{\lambda^{ch}}$. Ces saturations entraînent une perte de signifiance dans les variables internes de l'algorithme de décodage turbo (bloc SISO-1 et SISO-2), soit donc une augmentation du bruit de quantification ce qui se traduit par un aplatissement des courbes de taux d'erreur. Par conséquent, nous sélectionnons la valeur $A_{\lambda^{ch}} = 10$ pour la suite de l'analyse.

Il convient de préciser par ailleurs qu'une telle analyse de l'influence de la représentation en virgule fixe d'un turbo décodeur constitue un travail original car ce type d'analyse n'est pas présent dans la littérature scientifique. En effet, les travaux disponibles dans la littérature sont limités par des schémas "classiques" de quantification qui conduisent à un pas très grossier lorsque la dynamique $A_{\lambda^{ch}}$ est contrainte à une puissance de deux.

La figure 4.9 représente les performances FER pour différentes valeurs des paramètres en virgule fixe définies dans la section de modélisation en virgule fixe du décodeur et exprimées par le triplet $(N_{\lambda^{ch}}, N_\alpha, N_A)$.

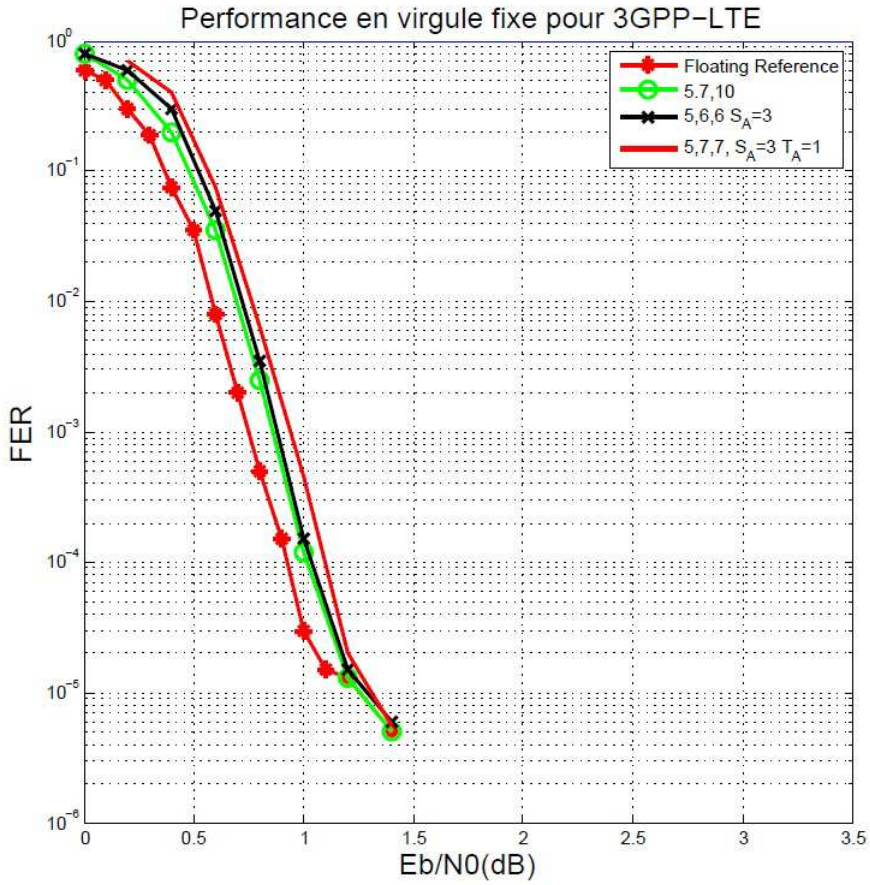

FIGURE 4.9 – Analyse de la performance de l'algorithme de turbo-décodage en virgule fixe

D'après la figure 4.9, il est clair qu'aucune perte n'est observée en saturant 3 bits sur les

Signal	Configuration A	Configuration B
LLR Canal *a priori*	(10,5)	(10,5)
Métriques de branches (RCS $r = 1/2$)	(20,6)	(20,6)
Métriques de chemin	(40,7)	(20,6)
Les bits saturés pour α/β (S_α)	2	2
Les messages extrinsèques *a posteriori*	(40,7)	(20,6)
Nombre de bits saturés sur les messages extrinsèques (S_A)	3	3

TABLE 4.2 – Les paramètres virgule fixe optimaux du turbo-décodeur

fiabilités extrinsèques ($S_A = 3$) lorsque $N_\alpha = 3$, tandis que la saturation de seulement 1 bit sur les récursions métriques ($S_\alpha = 1$, $N_\alpha = 6$) conduit à une perte de l'ordre de 0.18dB pour les forts SNR. La troncature d'un bit des itérations métriques ($T_\alpha = 1$) et les fiabilités extrinsèques ($T_A = 1$) dégrade légèrement les performances FER pour les faibles SNR où les LSBs portent la plupart de l'information, ce qui n'est pas surprenant, tandis que les pertes deviennent presque négligeables pour les forts SNR. Finalement, visant à une solution à très faible complexité, les itérations et les métriques extrinsèques peuvent être représentées sur $N_\alpha = N_A = 6$ bits avec une perte d'implémentation de 0.3 dB pour les forts SNR.

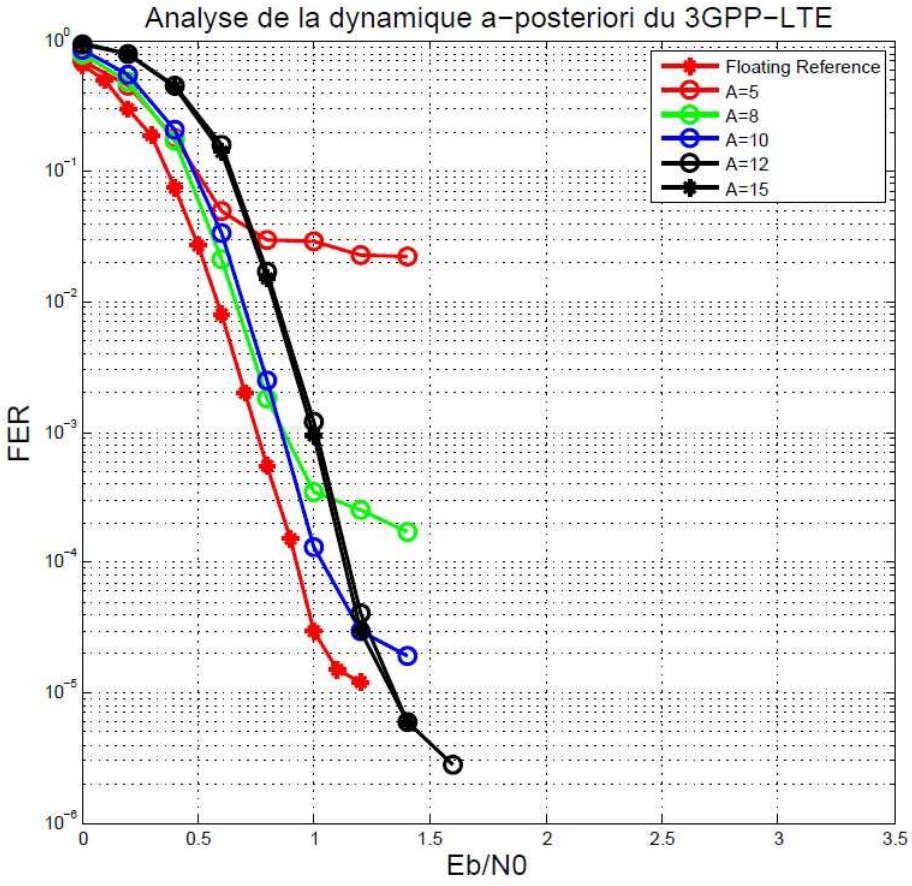

FIGURE 4.10 – Performance virgule fixe de bout en bout pour différentes valeurs de $A_{\lambda^{ch}}$

Le tableau 4.2 résume les paramètres de la spécification en figure fixe obtenue dans deux configurations. La première configuration, qui sera par la suite nommée configuration A, est orientée pour obtenir les meilleures performances, tandis que la deuxième configuration , qui sera par la suite nommée configuration B, est dédiée pour une faible complexité.

La configuration B est plus impliquée pour analyser l'effet conjoint de la dynamique des entrées et la quantification grossière de l'algorithme sur la performance totale de correction d'erreur. La

figure 4.10 montre que les courbes de FER s'aplatissent pour $A_{\lambda^{ch}} = 5$ et $A_{\lambda^{ch}} = 8$. Ceci est dû à la distorsion importante introduite par une très forte saturation des entrées *a priori* du canal, mais également au bruit de quantification généré en propre par l'algorithme (blocs SISO-1 et SISO-2). La valeur $A_{\lambda^{ch}} = 10$ conduit à une perte minimale de performances par rapport à la courbe de référence obtenue avec l'arithmétique à virgule flottante, mais en contrepartie il apparaît une saturation des performances pour de forts SNRs (aplatissement de la courbe $FER = f(Eb/N0)$). *A contrario*, l'utilisation de $A = 12$ empêche ce phénomène d'aplatissement, mais ceci est obtenu au prix d'une perte en SNR de l'ordre de $0.25dB$ par rapport à la courbe de référence (i.e. virgule flottante).

4.3 Optimisation du décodage LDPC

Dans cette section, nous présentons un modèle en virgule fixe du décodeur LDPC pour aboutir à une analyse des performances FER en virgule fixe. Nous commençons par présenter le principe du décodage LDPC. Ensuite, nous présentons le modèle en virgule fixe des différentes opérations impliquées dans l'algorithme de décodage, puis nous analysons les performances de l'approche proposée en fonction du niveau de bruit introduit par le canal de transmission et des différents paramétrages possibles pour la représentation en virgule fixe de l'algorithme. Avant de présenter l'algorithme de décodage LDPC, nous présentons tout d'abord le principe général des codes LDPC pour bien expliciter ensuite le principe de leur décodage.

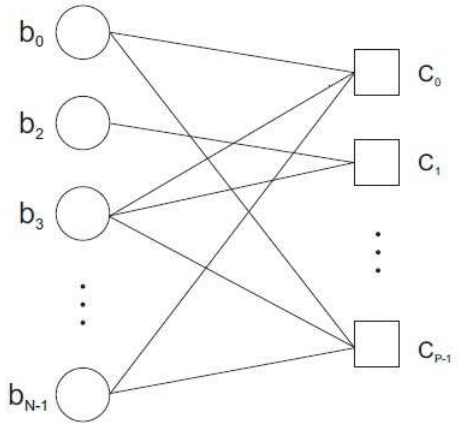

FIGURE 4.11 – Exemple du graphe tanneur

Les codes LDPC

Les codes LDPC sont des codes linéaires définis par une matrice creuse H appelée matrice de contrôle de parité. On dit qu'un mot de code x est valide s'il appartient à l'espace nul ou au noyau de H défini par l'ensemble H ($Hx^T = 0$). La matrice $H = \{h_{ij}\}$ de contrôle de parité possède un nombre de colonnes N égal au nombre de bits dans le mot code transmis et un nombre de lignes M correspondant au nombre de contraintes de contrôle de parité. La différence $P = N - M$ désigne le nombre de bits de parité ajoutés par le codeur LDPC. Chaque ligne de la matrice décrit une contrainte de contrôle de parité. Lorsque le j^{eme} bit du mot de code participe au $i^{ème}$ contrôle de parité, alors le terme $h_{i,j}$ de la matrice H est fixé à l'unité. Dans le contraire, il est nul.

Les codes LDPC peuvent être également décrits en utilisant un graphe appelé graphe de Tanner [89] qui est composé de nœuds de variables (VNs), représentés avec des cercles et de nœuds de contrôle (CNs) représentés par des carrés. Chaque VN représente un bit du mot code transmis et correspond à une colonne de H. Une connexion entre le $i^{ème}$ nœud variable VN et le $j^{ème}$ nœud de contrôle CN, dénommée *branche*, correspond à un terme $h_{ij} = 1$ dans la matrice

H de contrôle de parité et relie graphiquement la contrainte de parité à un bit du mot code. Le nombre des branches connectées à un VN (CN) est appelé le degré de nœud de variable, d_v (degré de nœuds de contrôle, d_c). Un exemple de graphe de Tanner est présenté par la figure 4.11.

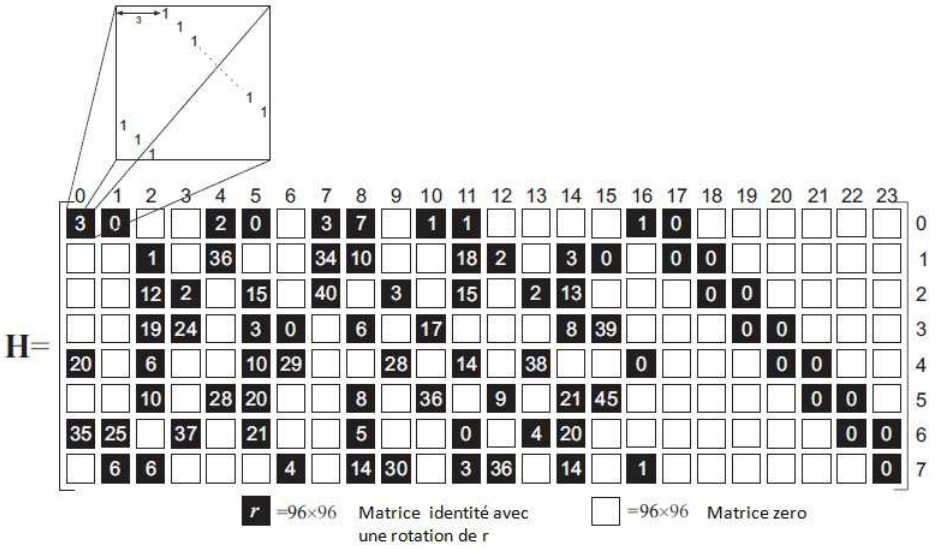

FIGURE 4.12 – Matrice prototype du LDPC WIMAX 2/3a

La conception de la matrice de contrôle de parité affecte la performance de correction d'erreur et la complexité du décodeur LDPC [38][67]. L'idée de base consiste à arranger les '1' dans la matrice de contrôle de parité selon des configurations qui facilitent la mise en parallèle des opérations de décodage. Ainsi, la matrice de contrôle de parité est partitionnée en matrices carrées plus petites, qui peuvent être des permutations ou des changements cycliques de la matrice unité, encore appelée matrice circulante [95]. La figure 4.12 montre la matrice prototype du code LDPC WiMAX 2/3a avec une longueur 2304 : elle est partitionnée en sous-matrices de taille $Z \times Z$ avec $Z = 96$, où une entrée nulle correspond à toutes les matrices 0, et une entrée non nulle spécifie la rotation appliquée à la matrice unité.

Décodage des codes LDPC

L'algorithme de décodage des codes LDPC est communément nommé algorithme de "propagation de la croyance" ou BP (pour *Belief Propagation*) ou plus généralement l'algorithme de passage de messages [79]. Cet algorithme est optimal si le code ne contient pas de cycles mais il peut cependant être considéré en pratique comme une référence pour des codes comportant des cycles. Dans ce cas, la séquence des élaborations, nommée ordonnancement (*schedule*), affecte considérablement la performance en termes de vitesse de convergence et de taux de correction d'erreur.

L'ordonnancement le plus simple est l'ordonnancement à deux phases FS (*flooding Schedule*) [41] qui se déroule en deux phases consécutives où tous les nœuds de contrôle de parité et tous les nœuds de variables sont mis à jour dans un ordre pré-déterminé.

Un ordonnancement plus intéressant mais aussi plus efficace concerne l'ordonnancement à couche (ou LS pour *Layered Schedule*) [67][41]. Comparé à la variante à deux phases (FS), l'ordonnancement de type LS double pratiquement la vitesse de convergence de décodage pour des codes comportant des cycles ou n'en comportant pas [58]. Ceci est atteint en regardant le code comme la connexion de couches, "*layers*" [49] qui échangent des messages de fiabilité. Spécifiquement, des messages a posteriori sont disponibles pour la couche suivante immédiatement après le calcul, et non pas à l'itération suivante comme dans la phase FS. Les couches peuvent être des ensembles de nœuds de contrôles (CN) ou de nœuds de variables (VN) consécutifs et, par

conséquent, l'algorithme basé sur CN (ou horizontal) ou l'algorithme basé sur VN (ou verticale) ont été définis dans [41].

Décodage horizontal des couches HLD (Horizontal Layered Decoding) L'algorithme HLD met à jour les contraintes de contrôle de parité séquentiellement autour de la matrice de contrôle de parité. La principale caractéristique de cet algorithme est la mise à jour continue, lors du décodage, d'une métrique cumulative y_n associée à chaque nœud de variable (VN) dans le code, $n = 0, 1, ..., N-1$, et nommée sortie souple (ou SO pour *Soft Output*).

La mise à jour des informations disponibles au nœud de contrôle numéro m (qui sera noté CNm dans la suite) , avec $m = 0, 1, ..., M-1$, est basée sur la disponibilité de messages vtoc (pour *Variable-To-Check*), notés $\mu_{n,m}$, dirigés du nœud variable (VN) numéro n (qui sera noté par suite VNn) vers le nœud CNm, et mis à jour selon :

$$\mu_{m,n}^{(q)} = y_n^{(q)} - \epsilon_{n,m}^{(q)} \tag{4.27}$$

où $\epsilon_{n,m}^{(q)}$ désigne le message ctov (pour *check-to-variable*) propagé du nœud CNm vers le nœud VNn au cours de l'itération précédente. Dans l'équation (4.27), l'indice n appartient à l'ensemble N_m constitué des indices des nœuds VNs connectés au noeud CNm, et l'indice $q = 0, 1, ..., N_{it,max} - 1$ désigne l'indice de l'itération.

Les messages raffinés *ctov* $\epsilon_{n,m}^{(q+1)}$ sont produits comme résultat de la mise à jour des nœuds de contrôle (*check node*). Et en se basant sur ceci, l'ensemble des sorties souples SOs impliqués dans le nœud de contrôle CN m, c'est à dire y_n avec $n \in N_m$, est mis à jour par la relation suivante :

$$y_n^{(q+1)} = \mu_{m,n}^{(q)} + \epsilon_{m,n}^{(q+1)} = y_n^{(q)} - \epsilon_{n,m}^{(q)} + \epsilon_{m,n}^{(q+1)}. \tag{4.28}$$

Grâce au mécanisme décrit par les relations (4.27) et (4.28), les opérations de contrôle des nœuds comptent sur les mises à jour des sorties souples SOs. L'algorithme HLD est initialisé à l'itération $q = 0$ avec :

$$\begin{cases} y_n^{(0)} = \lambda_n^{ch} \\ \epsilon_{m,n}^{(0)} = 0 \end{cases} \tag{4.29}$$

avec λ_n^{ch} est la LLR de l'estimation du canal *a priori* des bits reçus dans le bruit, $m = 0, 1, ..., M-1$ et $n \in N_m$.

Mise à jour des nœuds de contrôle : Concernant la mise à jour de contrôle de nœud, il a été montré dans [67] que la contrainte de contrôle de parité peut être vue comme un code convolutif à deux états, où un état (S_0) est associé à la parité paire, et l'autre état (S_1) à la parité impaire. La taille du bloc du code équivalent est égale au degré d_c des nœuds de contrôle (CN). Un exemple de représentation en treillis est donné par la figure 4.13.

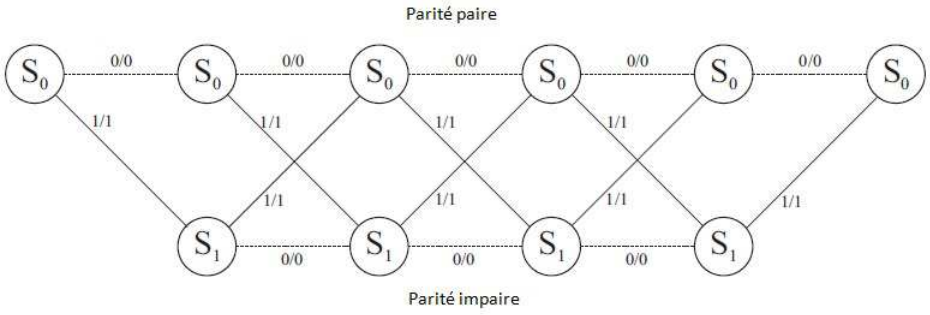

FIGURE 4.13 – Représentation du treillis à deux états d'une contrainte de vérification de parité avec $d_c = 5$

Cette analogie permet à l'algorithme BCJR d'être également utilisé pour la mise à jour du contrôle de parité des codes LDPC, et l'algorithme de décodage résultant est dénommé TDMP (pour *Turbo-Decoding Message Passing*) [67]. L'algorithme est alimenté avec des messages vtoc comme information *a priori* et produit des messages ctov comme sorties *a posteriori*, sans métrique branche du canal. Donc, dans la mise à jour du nœud de contrôle CN m, les métriques cumulées de chemin sont simplifiées selon :

$$\begin{cases} \alpha_{k+1} = max^*(\alpha_k, \mu_{m,n}^{(q)}) - max^*(\alpha_k + \mu_{m,n}^{(q)}, 0) \\ \beta_k = max^*(\beta_{k+1}, \mu_{m,n}^{(q)}) - max^*(\beta_{k+1} + \mu_{m,n}^{(q)}, 0) \end{cases} \tag{4.30}$$

où le paramètre $k = 1, 2, ..., d_c(m) - 1$ désigne l'indice temporel au sein du treillis, avec $d_c(m)$ le degré du nœud de contrôle CNm, et $n = N_m(k)$ l'indice du nœud de variable VN impliqué à l'étape k. Les relations de mise à jour données par (4.30) sont initialisées avec $\alpha_0 = 1$ et $\beta_{d_c} = 1$.

Par la suite, le calcul des informations extrinsèques *a posteriori* peut être reformulé de la manière suivante :

$$\epsilon_{m,n}^{(q+1)} = max^*(\alpha_k, \beta_{k+1}) - max^*(\alpha_k + \beta_{k+1}, 0) \tag{4.31}$$

avec $k = 0, 1, ..., d_c(m) - 1$ et $n = N_m(k)$.

4.3.1 Modélisation et *design* en virgule fixe d'un décodeur LDPC

Comme dans le cas du turbo décodeur, la modélisation en virgule fixe est fondée sur une approche en cascade où les opérations impliquées dans l'algorithme de décodage sont quantifiées les unes après les autres. Nous pouvons donc appliquer le modèle préalablement décrit que nous proposons dans le cadre de cette thèse. Il permet d'analyser indépendamment l'effet sur les performances (a) des entrées LLR *a priori*, (b) des messages ctov, et (c) des métriques cumulées de chemins au sein des relations de mise à jour des nœuds de contrôle.

1-Calcul des messages vtoc : Le calcul des messages vtoc μ selon (4.27) implique les sorties souples SOs et les messages ctov. Considérons que les entrées LLR sont quantifiées avec (A_λ, N_λ) et les messages ctov sont quantifiés avec (A_ϵ, N_ϵ) et soient Δ_λ et Δ_ϵ leur résolution, respectivement. Soit le rapport $\rho = \Delta_\lambda/\Delta_\epsilon$ contraint à être une puissance de deux. Cette supposition réduit le nombre des variables indépendantes dans le modèle : seulement trois variables sont indépendantes parmi les quatre variables A_λ,N_λ,A_ϵ et N_ϵ indépendantes. C'est aussi la seule solution viable pour une implémentation optimisée de l'algorithme.

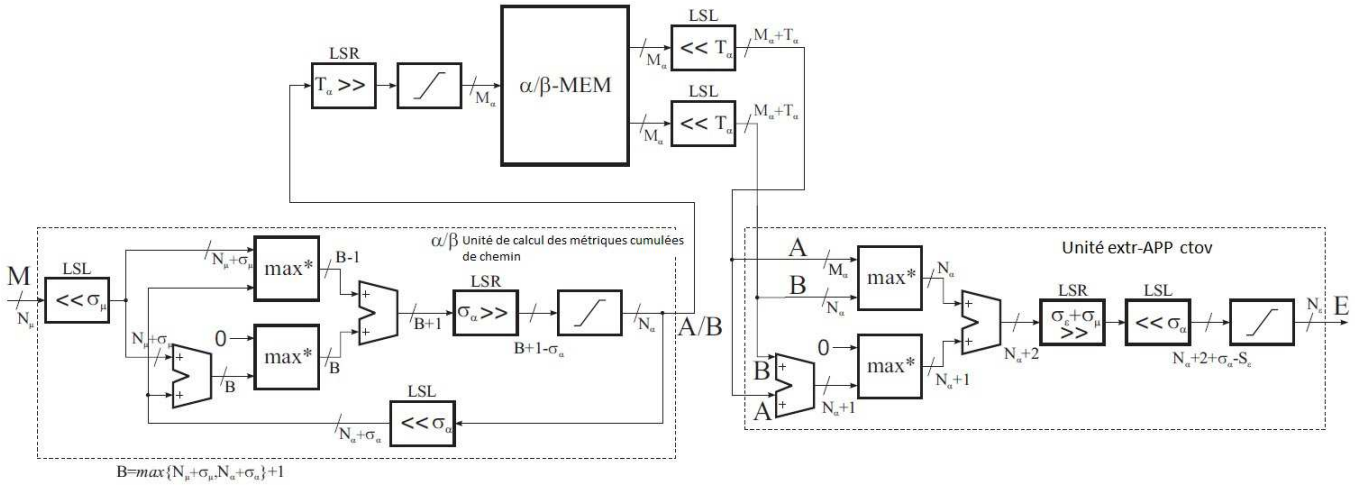

FIGURE 4.14 – Modèle virgule fixe du décodeur BCJR à 2-états

Si le rapport ρ est supérieur à 1, i.e. lorsqu'une résolution plus fine est utilisée pour les messages ctov, les rapports de vraisemblance canal (LLRs) *a priori* doivent être décalées à gauche de $\sigma_\lambda = log_2(\rho)$ bits à apporter sur la même résolution des messages ctov, ces derniers n'étant pas décalés ($\sigma_\epsilon = 0$). Dans l'autre cas, aucun décalage n'est nécessaire sur les entrées LLRs ($\sigma_\epsilon = 0$), tandis que les messages ctov doivent être décalés de $\sigma_\epsilon = -log_2(\rho)$ bits.

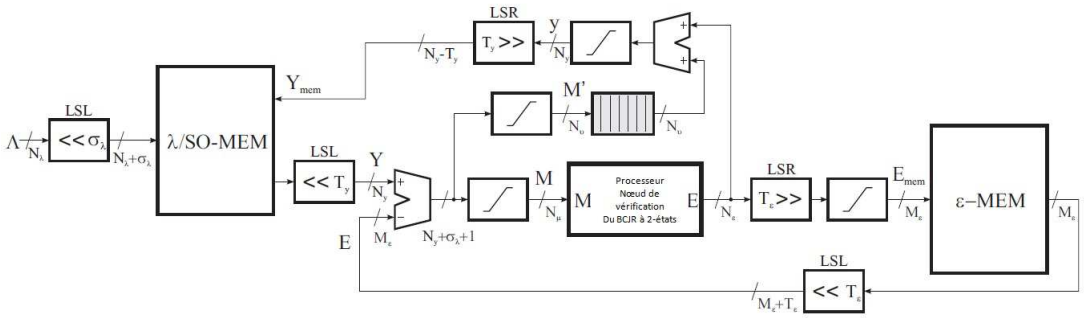

FIGURE 4.15 – Modèle virgule fixe du décodage à couche des codes LDPC

Les rapports de vraisemblance canal (LLRs) *a priori* sont utilisés pour initialiser les entrées souples (SO) et donc ces deux quantités possèderont la même résolution $\Delta_y = \Delta_\lambda$. Par conséquent, la relation entre la résolution (Δ_ϵ) des messages ctov et celle (Δ_y) des entrées souples (SO) sera identique à celle existant entre Δ_ϵ et Δ_λ, c'est à dire entre la résolution des messages ctov et celle des entrées LLR.

Sous ces considérations, les entrées souples (SOs) sont initialisées avec une version réduite des entrées LLRs de telle sorte que les données enregistrées ou récupérées du bloc mémoire du SO/λ utilisent la même résolution des messages ctov comme illustré dans la figure 4.15 .

Ainsi, pour calculer les messages vtoc en virgule fixe, il est possible dans ces conditions de réaliser directement la soustraction $Y - E$ et car les résolutions des deux opérandes sont identiques. Une fois les messages vtoc disponibles, il sont saturés de deux façons en parallèle : (a) une saturation sur N_μ bits pour l'entrée de l'unité de mise à jour des nœuds de contrôle (CN), et (b) une saturation sur N_v bits pour la mise à jour de la sortie souple (SO) dans la relation (4.28).

2- La mise à jour des SOs : La somme dans (4.28) est effectuée entre les messages ctov mis à jour E et les messages vtoc $M^{'}$ saturés sur N_v bits. Sa sortie est saturée sur N_y bits. Comme la sortie souple (SO) est toujours égale à la somme de d_v des messages ctov entrant à un nœud de variable (VN) donné, N_y est donné par :

$$N_y = N_\epsilon + \lceil log_2(d_{v,max}) \rceil \tag{4.32}$$

où $d_{v,max}$ représente le degré maximal du nœud de variable (VN) dans le code. Par contre, on parle d'une solution à faible complexité lorsque les SOs sont saturées sur un nombre de bits inférieur à N_y donné par (4.32).

3-Les récursions métrique d'état : L'expression (4.30) combine les messages vtoc avec les métriques de chemins (*forward* et *backward*), et de la même manière pour le calcul des messages vtoc. Différentes résolutions pour les deux signaux peuvent être considérées.

De même que précédemment, nous contraignons le rapport $\rho = \Delta_\lambda/\Delta_\epsilon$ à être égal à une puissance entière de deux. Ce facteur est utilisé pour aligner les représentations en virgule fixe M et A des variables μ et α représentant les métriques de chemins. Par conséquent, les représentations en virgule fixe M et A sont décalées respectivement de $\sigma_\mu = log_2(\rho)$ et $\sigma_\alpha = -log_2(\rho)$ lorsque le paramètre ρ est supérieur à l'unité. Dans le cas contraire, ces représentations ne sont

pas modifiées. Ainsi, la somme $\alpha + \mu$ dans (4.30) devient en représentation en virgule fixe :

$$A.2^{\sigma_\alpha} + M.2^{\sigma_\mu} \tag{4.33}$$

comme représenté sur la figure 4.14.

Les autres opérations de l'algorithme peuvent être représentées au format virgule fixe en utilisant la méthodologie qui a précédemment été décrite dans la section dédiée au turbo décodage. Seules quelques modifications peuvent être apportées afin de simplifier la mise en œuvre. Ainsi, comme indiqué sur la figure 4.14, en définissant le paramètre $B = max\{N_\alpha + \sigma_\alpha, N_\mu + \sigma_\mu\}$, à chaque itération nous pouvons représenter la nouvelle valeur de A en virgule fixe sur $B + 1$ bits. Cette valeur est tout d'abord décalée à droite de sa bits puis saturée sur le nombre désiré de N_α bits.

4-Les messages APP ctov : Comme montré dans la figure 4.14, les messages ctov sont calculés avec les métriques cumulées de chemin récursions disponibles au sein de la mémoire. Ces métriques sont représentées sur M_α bits ce qui implique que la mise à jour des messages raffinés ctov donnée par la relation (4.31) peut être représentée en virgule fixe sur $M_\alpha + 2$ éléments binaires. Une fois que l'information *a posteriori* est mise à jour, une opération de décalage (à gauche pour σ_α et à droite pour σ_ϵ) est réalisée afin de revenir à une résolution correspondant à celle des messages ctov. La saturation finale restaure la représentation sur N_ϵ bits.

4.3.2 Réduction de la taille de mémoire

Comme dans le cas du turbo décodeur, l'implémentation est fondée sur l'utilisation étendue de mémoire pour accumuler l'estimation en sortie et stocker les résultats intermédiaires correspondant à la mise à jour des métriques cumulées de chemin et des messages ctov.

Dans le cas du décodeur LDPC, la troncature est effectuée sur les messages ctov (T_ϵ bits), sur les SOs (T_y bits) et sur les métriques cumulées de chemin *forward* et *backward* (T_α bits). Comme indiqué sur la figure 4.14, ces signaux sont décalés (à gauche) juste après récupération de la mémoire, puis les saturations sont effectuées sur les messages ctov (saturés sur M_ϵ bits) et les récursions de métriques de chemins α/β (saturées sur M_α bits), tandis que les sorties souples SOs ne nécessitent pas de saturation après leur calcul.

4.3.3 Performance en virgule fixe du décodeur LDPC

Nous présentons dans ce paragraphe, les résultats obtenus avec la méthode de représentation en virgule fixe exposée précédemment pour un algorithme de décodage LDPC. Pour ces expérimentations, le code LDPC utilisé correspond à une version proche de celui utilisé par la norme WIMAX dont les paramètres principaux sont : un rendement de codage $R = 2/3$, $N = 1056$, un degré maximal $d_{v,max} = 6$ pour les nœuds de variable VN, et un degré maximal $d_{c,max} = 11$ pour les nœuds de contrôle CN .

L'effet de la quantification des entrées LLR est analysé sur la figure 4.16 où le taux d'erreur par paquet (FER) est représenté en fonction de la dynamique A_λ pour trois différentes valeurs de N_λ soit 4, 5 et 6 bits. Les quatre courbes représentées sur cette figure ont été obtenues pour un rapport signal à bruit $E_b/N_0 = 3.25dB$ à l'entrée du récepteur. La courbe correspondant à la représentation en virgule flottante sera prise comme référence. Nous pouvons constater que, pour des valeurs de N_λ de 4 ou 5 bits, les meilleures performances sont alors obtenues pour une dynamique A_λ de 10 bits. Par contre, lorsque N_λ est égal à 6 bits, la valeur optimale de la dynamique de A_λ est de 12 bits. Dans la suite, nous sélectionnons une précision des entrées LLR correspondant au réglage suivant $(A_\lambda, N_\lambda) = (10, 5)$ ce qui conduit à un bon compromis performances/complexité. Une analyse similaire a été menée pour la représentation en virgule fixe des métriques cumulées de chemins dans le processeur CN. Les résultats correspondants sont présentés sur la figure 4.17.

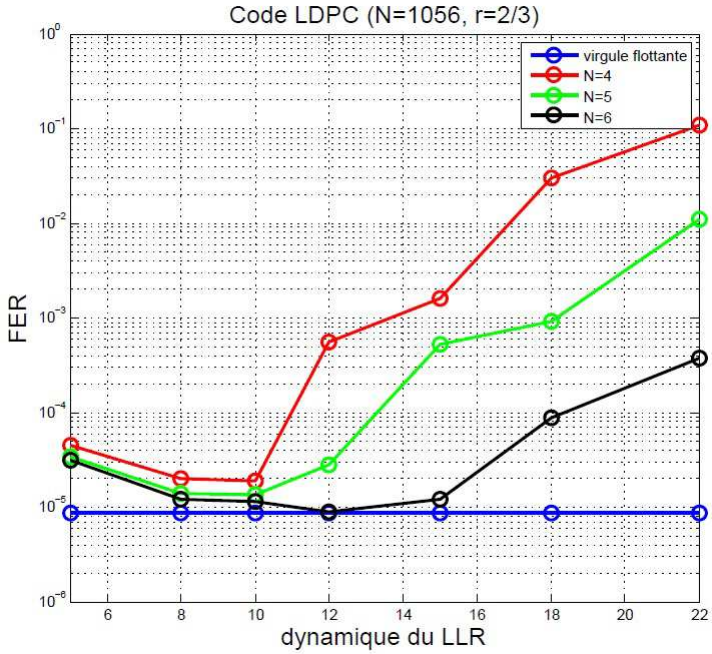

FIGURE 4.16 – Effet de la quantification des entrées LLRs

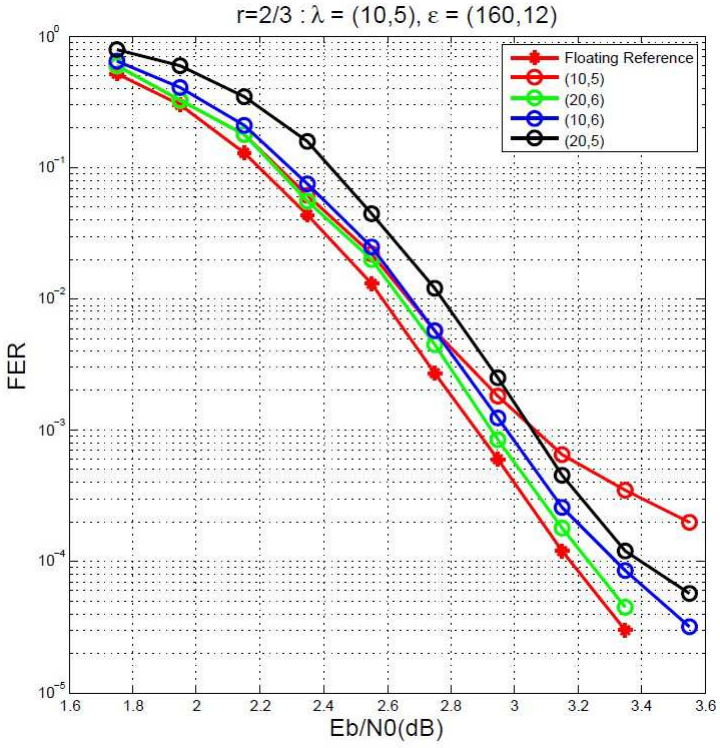

FIGURE 4.17 – Effet de la quantification des métriques cumulées de chemin dans le processeur CN

Dans ce cas, les courbes de performances (FER) sont évaluées en fonction de E_b/N_0, i.e. le rapport signal à bruit présent en entrée du filtre de réception. Afin d'éviter les pertes dues à la quantification des messages ctov et des sorties souples (SOs), une quantification très fine est utilisée pour ces deux signaux en se basant sur un nombre suffisant de bits et une dynamique

large.

En particulier, la précision (160,12) est utilisée pour les messages ctov tandis que 3 bits additionnels ($d_{v,max} = 6$) sont alloués pour les sorties souples SOs et $N_y = 15$. De même, pour les messages vtoc nous avons choisi de travailler avec une dynamique de $N_\mu = 15$ bits. Le reste de l'algorithme est exécuté en précision quasi-virgule flottante. Nous pouvons constater qu'il est suffisant d'utiliser environ 7 ou 8 bits afin d'éviter le phénomène de FER résiduel (i.e. aplatissement des courbes FER) et $N_\alpha = 6$ fournit de bonnes performances.

4.4 Conclusion

Nous avons abordé dans ce chapitre la problématique de l'optimisation des largeurs de données impliquées dans les algorithmes de système de traitement de signal ainsi que les opérations effectuées dans ces algorithmes. Nous avons commencé par proposer une modélisation en virgule fixe du turbo décodeur en se basant sur le principe de fonctionnement de l'opération de décodage. Cette modélisation est fondée sur la quantification en cascade des différentes opérations récursives dans l'algorithme de décodage. Une analyse des performances du modèle en virgule fixe est présentée sous forme de courbes de taux d'erreur par paquet FER en fonction du SNR. De plus, cette modélisation nous permet de faire une exploration totale du compromis performance/complexité de l'architecture. La même stratégie a été effectuée pour modéliser en virgule fixe le décodeur LDPC en analysant les performances des choix en virgule fixe par le biais des courbes FER en fonction des paramètres virgule fixe choisis en visant des solutions à faible complexité d'implémentation avec le minimum de pertes de performance.

Conclusion et Perspectives

L'utilisation de l'arithmétique en virgule fixe [14] pour implémenter les algorithmes de traitement numérique de signal afin de pouvoir satisfaire aux contraintes de coût, de performance et de consommation d'énergie. La réduction du temps de mise sur le marché de ces applications nécessite l'utilisation d'outils de haut niveau de conversion automatique de l'arithmétique virgule flottante vers l'arithmétique virgule fixe. Cette conversion vise à minimiser les coûts de l'implémentation en termes de surface d'architecture, de consommation d'énergie et du temps d'exécution du code sous une contrainte de la précision des calculs pour garantir l'intégrité de l'application. Par conséquent, l'évaluation de la précision en arithmétique virgule fixe est une étape importante et nécessaire pour accomplir la conversion automatique en virgule fixe au sein de ces outils et aboutir à optimiser la largeurs de données en format virgule fixe. Ainsi, à chaque itération du processus d'optimisation, la précision est évaluée à travers des métriques. Cette évaluation peut se faire par des simulations en format virgule fixe qui conduisent à des temps d'évaluation prohibitifs ou bien en proposant des modèles analytiques permettant de réduire d'une façon très significative les temps d'évaluation. L'objectif de cette thèse est de proposer des modèles analytiques pour évaluer la précision des systèmes de communication numérique et de traitement numérique de signal formés d'opérateurs non lisses et non linéaires en terme du bruit.

Le travail de recherche effectué au cours de cette thèse est fondé sur trois parties différentes. Dans un premier temps, nous avons défini des modèles analytiques d'évaluation de la précision pour les opérateurs de décision et la cascade d'opérateurs de décision. Cette étude a porté sur la caractérisation de la propagation des erreurs de décision dues à la quantification des données au sein d'une cascade d'opérateurs non lisses. Nous avons ainsi proposé un modèle analytique permettant d'estimer la probabilité d'erreur de décision au niveau de chaque élément de la cascade. Cette approche peut être appliquée à tout type de système constitué de cascade d'opérateurs décision. A titre d'exemple, nous avons appliqué l'approche proposée à l'évaluation des performances de l'algorithme de décodage sphérique SSFE en arithmétique virgule fixe. Nous avons ainsi montré que le modèle analytique permet d'estimer les probabilités d'erreur de décision pour chaque opérateur de décision impliqué dans la cascade de cette application.

La deuxième partie concerne l'évaluation de la précision des systèmes itératifs d'opérateurs non lisses de décision, en particulier, les algorithmes d'égalisation à retour de décision. Nous avons tout d'abord caractérisé le phénomène de propagation des erreurs de décision au sein de ces systèmes. Deux modèles analytiques ont été proposés afin d'évaluer la probabilité d'erreur de décision de telles structures récursives La première approche basée sur la résolution d'un système non linéaire par le biais de la méthode de Newton-Raphson donne une estimation fiable de la probabilité d'erreur de décision due à la quantification en format virgule fixe. La seconde approche fournit une borne supérieure de la probabilité d'erreur de décision avec un temps d'évaluation largement réduit par rapport à la première approche. Cette dernière a été appliquée pour estimer la précision d'un égaliseur à retour de décision DFE. Une solution a été proposée pour étendre cette approche pour l'évaluation du DFE dans sa version adaptative. La dernière étape de ce travail a été consacrée à l'optimisation de la largeur des données impliquées dans l'algorithme DFE pour réaliser une implémentation efficace sur FPGA en termes de consommation de ressources et de puissance sous une contrainte de précision exprimée en fonction de la probabilité d'erreur de décision.

La dernière partie de cette thèse s'articule autour de la modélisation en arithmétique virgule

fixe des algorithmes de décodage itératif de type turbo-décodage et décodage LDPC. Cette étude a conduit à la proposition de deux modèles en virgule fixe pour le turbo-décodeur et décodeur LDPC en quantifiant les différentes opérations et récursions impliquées dans le fonctionnement des algorithmes de décodage. La loi de quantification utilisée se base sur une représentation différente du schéma classique utilisant le nombre de bits pour la partie entière et le nombre de bits pour la partie fractionnaire. La représentation proposée accorde davantage de flexibilité pour la dynamique des signaux qui peut prendre des valeurs différentes d'une puissance de deux. Nous avons ainsi proposé plusieurs paramétrages réalisant différents compromis entre performances de l'application en arithmétique virgule fixe (taux d'erreur par paquet ou FER) et complexité de l'architecture. Des techniques de saturation et de troncature ont été proposées pour réduire la taille des mémoires afin de viser à des solutions à faible complexité. Des simulations de bout en bout ont permis de valider l'approche proposée en quantifiant les récursions impliquées dans les algorithmes de décodage pour en évaluer les performances des choix des largeurs des données en arithmétique virgule fixe.

Perspectives

La caractérisation des opérateurs non lisses est une étape obligatoire pour développer des modèles analytiques permettant l'évaluation de la précision des systèmes constitués de ce type d'opérateurs. Par conséquent, il s'avère important de pouvoir établir une classification de ce type d'opérateurs selon leurs propriétés mathématiques, et en particulier en fonction des propriétés de continuité et dérivabilité. Cette classification peut être la base de développement d'une approche générale permettant l'évaluation de la précision des systèmes de communication numérique et de traitement numérique de signal afin d'être en mesure de proposer un outil de haut niveau permettant l'évaluation de la précision de ces systèmes.

Une seconde perspective concerne l'implémentation en virgule fixe de l'égaliseur à retour de décision en visant à quantifier les opérations itératives impliquées dans cet algorithme. De plus nous pouvons également réduire la taille des mémoires pour réaliser une architecture en arithmétique virgule fixe à faible complexité. En addition, nous pouvons également étendre les approches analytiques proposées pour évaluer la précision du DFE, en tenant compte de la variation des coefficients des filtres *feedback* et *forward* à chaque itération pour augmenter la précision de l'évaluation et ceci en prévoyant cette variation ou bien en utilisant des techniques d'apprentissage non-supervisé pour estimer les bornes de la variation de ces coefficients et les intégrer dans les approches analytiques proposées.

Une dernière perspective peut être évoquée au sujet des algorithmes de décodage itératif. En effet, une approche analytique pour évaluer la précision de ces algorithmes peut être abordée en s'inspirant des modèles développés dans le cadre du DFE. En effet, les deux structures sont similaires puisqu'elle sont basées sur des calculs itératifs incluant un bloc de décision en sortie. Par conséquent, les modèles développés pour le cas du DFE dans le cadre de cette thèse peuvent être étendus pour évaluer la précision du turbo-décodeur et du décodeur LDPC selon les contraintes de précision et de coût d'implémentation souhaitées par l'utilisateur.

Annexe A

Quantification uniforme et quantification double

Nous présentons dans cet annexe le principe de la quantification uniforme et double avec caractérisation dans le domaine de la fonction caractéristique.

A.1 Quantification uniforme

Le processus de quantification est par définition non lisse dans le sens strict. Pour déterminer l'applicabilité du modèle PQN à l'étude du processus de quantification, la relation entre la dynamique D du signal d'entrée et le pas de quantification q doit être considérée. Lorsque le pas de quantification est beaucoup plus petit que la dynamique du signal ($q \ll \alpha$), il est possible de modéliser l'erreur introduite par la quantification en utilisant le modèle PQN. Cette hypothèse d'un faible pas de quantification (relativement à la dynamique du signal) est à l'origine des propriétés du modèle additif PQN.

Dans le cas d'un opérateur non lisse, le pas de quantification est large et souvent comparable à la dynamique du signal D. Dans ces conditions, les propriétés de l'erreur de quantification, y compris la forme de bruit, sa propriété d'additivité et son indépendance statistique avec le signal d'entrée sont discutables et remises en question. Il est alors nécessaire de considérer un cadre plus général où les propriétés statistiques du bruit de quantification varient en fonction de la FDP du signal d'entrée et en fonction du pas de quantification.

Dans cette section, nous présentons les dynamiques de la quantification des signaux réels dans le domaine de la fonction caractéristique. Cette analyse est ensuite appliquée pour les signaux quantifiés afin d'apprécier le processus de quantification durant une mise en cascade de deux opérateurs de quantification.

A.1.1 Quantification dans le domaine de la fonction caractéristique

Le phénomène de quantification ressemble à l'échantillonnage de Nyquist. De façon analogue à l'échantillonnage de Nyquist qui est défini sur l'axe temporel, la quantification peut être considérée comme l'échantillonnage tout au long de l'axe d'amplitude du signal. De ce fait, la FDP du signal et sa fonction caractéristique (FC) qui est sa transformée de Fourier sont analogues au signal dans son domaine temporel et sa représentation dans le domaine de Fourrier. De la même façon, le pas de quantification q joue un rôle similaire à la période d'échantillonnage de Nyquist T.

La FC $\phi_x(u)$ du signal original x se répète tout au long de l'axe de son domaine u, la fréquence radian de quantification a un intervalle $\psi = \frac{2\pi}{q}$. La figure A.1 montre le chevauchement des fonctions caractéristiques du signal quantifié lorsque le pas de quantification est supérieur ou égal à q_b. La largeur de bande d'un signal quantifié est définie par l'intervalle des pulsations $[0, W_x]$, où W_x correspond à la pulsation limite engendrant une discontinuité dite à *crenelage* entre les images successives de la fonction caractéristique. Cette largeur de bande est donc est déterminée

à partir du pas de quantification selon :

$$W_x = \frac{2\pi}{q_b} \quad \text{tel que} \quad \forall \quad |u| \geq \frac{2\pi}{q_b} \tag{A.1}$$

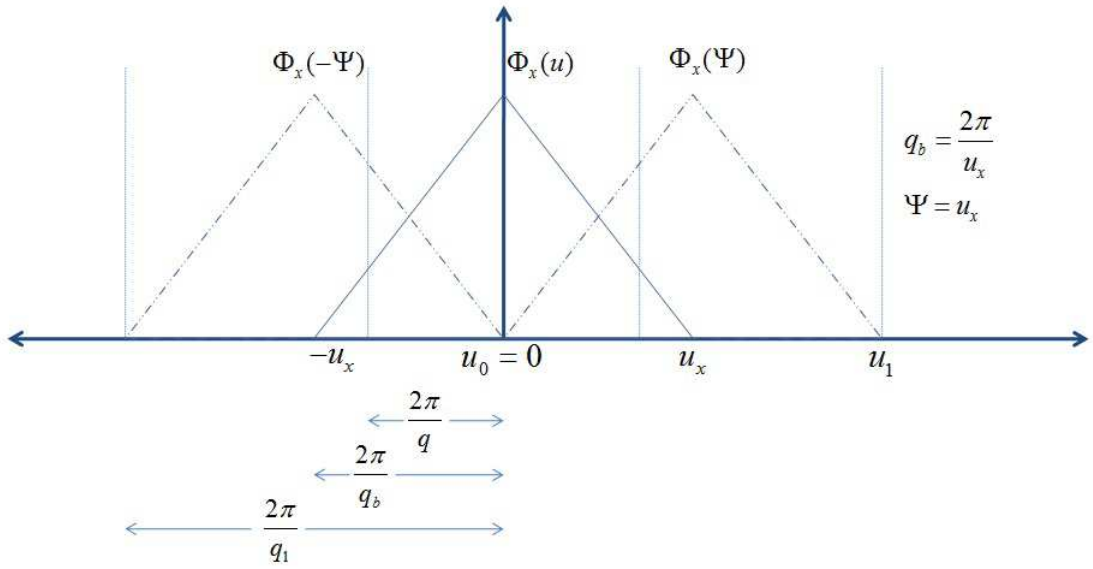

FIGURE A.1 – Domaine de la fonction caractéristique

Lorsque le pas de quantification est suffisamment faible, la fréquence angulaire est grande de telle sorte qu'elle ne permet pas le chevauchement de la fonction $\phi_x(u)$. Comme les pas de quantification augmentent, la répétition est plus fréquente provoquant un crénelage/ Cette observation est connue par le premier théorème de quantification (TQ1). En d'autres termes, ce n'est que l'application du théorème d'échantillonnage de Nyquist selon l'axe d'amplitude du signal. Dans la figure A.1 q_1 est le plus grand pas de quantification de telle sorte que TQ1 est satisfaite.

Le second théorème de quantification (TQ2) assouplit les conditions fixées par TQ1 et permet le chevauchement des images de répétition des FC du signal. Le second théorème de quantification (TQ2) précise les conditions dans lesquelles les moments du signal d'origine peuvent être récupérés obtenusA.1, le plus grand pas de quantification qui peut satisfaire à la condition de TQ2 est noté q_b. Soit S_r le moment d'ordre r du signal erreur de quantification et R_r est la covariance entre le moment d'ordre r de signal d'entrée x et l'erreur de quantification e. Les moments du signal quantifié $\hat{x}$ sont données par :

$$E[\hat{x}^r] = E[x^r] + S_r + R_r \qquad avec \qquad R_r = E[(x - \bar{x})^r (e - \bar{e})^r] \tag{A.2}$$

Lorsque les théorèmes TQ1 et TQ2 sont satisfaits, la valeur des coefficients R_r est nulle pour tout r. En d'autres termes, le modèle PQN peut être utilisé pour déterminer analytiquement l'effet du bruit de quantification. Le pas de quantification est assez grand pour que q_b marque la frontière entre l'applicabilité et la non-applicabilité du modèle PQN. Lorsque le pas de quantification est augmenté au-delà de cette limite, la valeur de R_r devient non nulle, et les propriétés statistiques du signal quantifié s'écartent des valeurs données par le modèle PQN.

Dans la pratique, le signal d'entrée prend un ensemble de valeurs finies et il est donc limité dans le domaine de la FDP. Théoriquement, cela rend la FC de la bande de signal illimitée. En d'autres termes, contrairement au cas représenté sur la figure A.1, la FC est en fait théoriquement nulle seulement à l'infini. Les théorèmes de quantification TQ3 et TQ4 se placent dans des conditions lorsque n'importe quel moment arbitraire du bruit de quantification et du signal peuvent être mutuellement orthogonales (i.e R_r dans l'équation (A.2) est nulle pour n'importe quel r) et le

bruit de quantification continue d'être uniformément distribuée indépendamment du lieu où la FC passe à 0. Par conséquent, le modèle PQN est applicable sur un quantificateur de pas de quantification q si la fonction caractéristique d'entrée et ses dérivées prennent la valeur 0 à tous les points correspondant aux multiples de la fréquence angulaire de quantification $\phi = \frac{2\pi}{q}$.

A.2 La quantification double

Tout système dans la pratique se compose de nombreux quantificateurs qui causent une quantification répétée tout au long d'un chemin de données en virgule fixe. Lorsque les théorèmes de quantification sont bien définis pour la première quantification, l'application du modèle PQN pour le phénomène de double quantification n'est pas rigoureusement définie car le sujet d'une cascade de plusieurs quantificateur n'est pas précisé par le modèle PQN particulièrement lorsqu'il s'agit d'opérateurs non lisses. Dans [85], il a été supposé que tous les quantificateurs sont toujours conformes aux conditions du modèle PQN. Afin de justifier cela, considérons une double quantification comme dans la figure A.2 avec deux quantificateurs en mode arrondi. Bien que la puissance du bruit total est affectée en raison d'une moyenne non nulle finie, la dynamique de la quantification n'est pas affectée dans le mode par troncature.

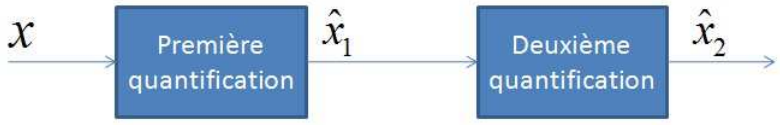

FIGURE A.2 – Quantification double

Dans les travaux présentés dans [92], la function densité de probabilité $f_{\hat{x}_1}(x)$ du signal $\hat{x}_1$ à la sortie du premier quantificateur est donnée par :

$$\begin{aligned} f_{\hat{x}_1}(x) &= (f_x(x) * f_{b_1}(x)).c_{q_1}(x) \\ &= \sum_{m=-\infty}^{\infty} \delta(x - m.q_1) \int_{m.q_1-\frac{q_1}{2}}^{m.q_1+\frac{q_1}{2}} f_x(\alpha)d\alpha. \end{aligned} \tag{A.3}$$

avec δ est la fonction d'impulsion, $f_x(x)$ et $f_{n_1}(x)$ sont respectivement les FDP du signal d'entrée x et du signal de quantification au niveau du premier quantificateur. $c_{q_1}(x)$ est le train d'impulsions du premier quantificateur espacé de q_1 unités et il est donné par :

$$c_{q_1}(x) = \sum_{m=-\infty}^{\infty} q_1.\delta(x - m.q_1). \tag{A.4}$$

L'entrée du second quantificateur $\hat{x}_1$ est de nature discrète. Cependant, les dynamiques de la quantification des signaux dont la FDP est discrète sont inchangées. En raison de l'amplitude discrète du signal d'entrée, il suffit de considérer l'erreur due à la quantification au niveau des points discrets. La FDP discrète du bruit de quantification au niveau du deuxième quantificateur $f_{b_2}(x)$ peut être écrite en discrétisant encore le niveau avec le pas de quantification du quantificateur précédent :

$$f_{b_2}(p) = \begin{cases} \frac{1}{k} & -\frac{k}{2} \leq p < \frac{k}{2} \\ 0 & \text{autre part} \end{cases} \tag{A.5}$$

avec $p = \frac{\hat{x}}{q_1}$ est une variable discrète correspondante à $\hat{x}$ et $k = \frac{q_2}{q_1}$ est le rapport des pas de quantification. Lorsque un circuit arithmétique binaire est utilisé, il convient de noter que les pas de quantification des quantifications successives sont des puissance de 2. En d'autres termes,

échantillonner le train d'impulsions du premier quantificateur a un taux de k donne $c_{q_2}(x)$. Sa version discrétisée peut être écrite comme suit :

$$c_{q_2}(p) = \sum_{m=-\infty}^{\infty} k.\delta(p - m.k). \tag{A.6}$$

La FDP du signal $\hat{x}_2$ est une fonction de la FDP discrète. Elle peut être simplifiée suivant :

$$\begin{aligned} f_{\hat{x}_2}(p) &= (f_{\hat{x}_1}(p) * f_{b_2}(p)).c_{q_2}(p) \\ &= \left(\sum_{\beta=-\infty}^{\infty} f_{b_2}(p-\beta).f_{\hat{x}_1}(\beta) \right).c_{q_2}(p) \\ &= \left(\sum_{\beta=p-\frac{k}{2}}^{p+\frac{k}{2}} \frac{1}{k}.f_{\hat{x}_1}(\beta) \right).c_{q_2}(p) \\ &= \left(\sum_{\beta=p-\frac{k}{2}}^{p+\frac{k}{2}} \frac{1}{k}.f_{\hat{x}_1}(\beta) \right). \sum_{m=-\infty}^{\infty} k.\delta(p - m.k) \\ &= \sum_{m=-\infty}^{\infty} \delta(p - m.k) \sum_{\beta=m.k-\frac{k}{2}}^{m.k+\frac{k}{2}} f_{\hat{x}_1}(\beta). \end{aligned} \tag{A.7}$$

La FDP du signal $\hat{x}_2$ est une fonction de la FDP discrète $f_{\hat{x}_1}(p)$ avec β est le compteur de k échantillons discrètes de la FDP du signal $\hat{x}_1$ centrée sur $m.k$ pour toute m. Elle peut être simplifiée comme une fonction du signal en précision infinie comme :

$$\begin{aligned} f_{\hat{x}_2}(\frac{x}{q_1}) &= \sum_{m=-\infty}^{\infty} \delta(\frac{x}{q_1} - m.\frac{q_2}{q_1}) \sum_{\frac{x}{q_1}=m.\frac{q_2}{q_1}-\frac{q_2}{2q_1}}^{m.\frac{q_2}{q_1}+\frac{q_2}{2q_1}} f_{\hat{x}_1}(\frac{x}{q_1}) \\ f_{\hat{x}_2}(x) &= \sum_{m=-\infty}^{\infty} \delta(x - m.q_2) \sum_{x=m.q_2-\frac{q_2}{2}}^{m.q_2+\frac{q_2}{2}} f_{\hat{x}_1}(x). \end{aligned} \tag{A.8}$$

Considérons maintenant qu'un seul quantificateur soit utilisé pour obtenir $\hat{x}_2$ avec un pas de quantification $q_2 = k.q_1$. La FDP du signal directement obtenu de l'équation A.3 s'écrit par la relation suivante :

$$f_{\hat{x}_2}(x) = \sum_{m=-\infty}^{\infty} \delta(x - mq_2) \int_{mq_2-\frac{q_2}{2}}^{mq_2+\frac{q_2}{2}} f_x(\alpha)d\alpha \tag{A.9}$$

Le pas de quantification du deuxième quantificateur est un entier multiple du pas de quantification du premier quantificateur. Ainsi, la superficie totale couverte par l'intégrale entre le niveau $(-\frac{q_2}{2}, \frac{q_2}{2})$ est aussi bonne que l'addition de k échantillons de la FDP dans l'intervalle. Cela se traduit par :

$$\int_{mq_2-\frac{q_2}{2}}^{mq_2+\frac{q_2}{2}} f_x(\alpha)d\alpha = \sum_{x=m.q_2-\frac{q_2}{2}}^{m.q_2+\frac{q_2}{2}} f_{\hat{x}_1}(x). \tag{A.10}$$

En comparant les relations (A.8) et (A.9), il est clair que la FDP du signal quantifié en cas de double quantification est la même que dans le cas de la seule quantification. Ce résultat signifie que les théorèmes de quantification sont applicables même dans le cas de la double de quantification comme si elle est appliquée sur le signal d'origine. En d'autres termes, le modèle PQN peut être appliqué pour estimer les statistiques globales du bruit de quantification à la sortie d'un quantificateur quelconque indépendamment du fait que le quantificateur agit sur un signal continu ou un signal déjà quantifié.

Annexe B

La méthode de Newton-Raphson pour la résolution des systèmes non linéaires

B.1 Les systèmes non linéaires

Le cas des systèmes non linéaires est beaucoup plus délicat que celui des des systèmes linéaires car les non-linéarités sont génératrices d'instabilités numériques fortes. Considérons l'exemple relativement simple du système suivant de deux équations à deux inconnues :

$$\begin{cases} f_1(x,y) = x^2 - (y^2-1)^2 = 0 \\ f_2(x,y) == (x^2-2)^2 - 2y^2 = 0 \end{cases} \tag{B.1}$$

L'équation $f_1(x,y) = 0$ correspond à une ligne courbe dans le plan xy ; de même pour l'équation $f_2(x,y) = 0$. Le problème qui se pose alors consiste à rechercher les intersections de ces lignes courbes. En outre, ces lignes courbes peuvent dans certains cas dégénérer en simples points, ou encore en zones continues à deux dimensions (un demi-plan par exemple). Le cas particulier des fonctions ci-dessus est illustré par le diagramme suivant :

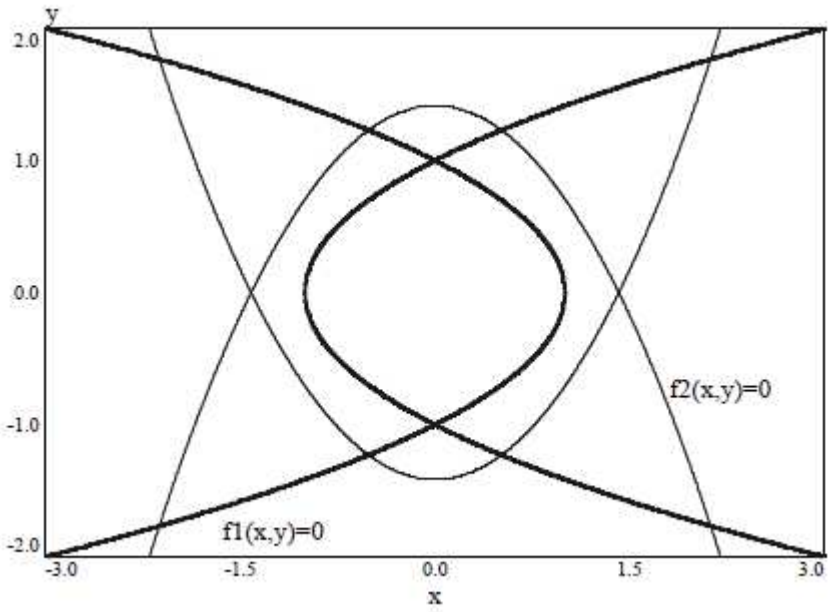

FIGURE B.1 – diagramme des cas particuliers

Les équations $f_1(x, y) = 0$ et $f_2(x, y) = 0$ définissent des coniques, en conséquence la résolution du système défini par ces 2 équations revient à chercher tous les points d'intersections entre ces coniques. Sans le cas de l'exemple précédent, on observe huit solutions possibles.

Si l'on généralise le problème précédent à un système de n équations à n inconnues, les lignes courbes précédentes sont généralement remplacées par des hyper-courbes de dimension n-1, ce qui rend le problème encore plus complexe. En fait, il n'y a pas de méthode miracle universelle et il est généralement nécessaire d'introduire des informations complémentaires sur le système à résoudre. Ceci peut concerner, par exemple, le nombre de solutions distinctes (en particulier, peut-on attendre une unique solution ?) ou bien encore la position approximative de ces solutions. Toute autre information *a priori* sur le système est généralement bienvenue, sous réserve de modifier en conséquence l'implémentation des méthodes numériques de résolution. Sous réserve que l'on parvienne à cerner le problème posé, on peut simplifier le problème et généraliser la méthode de Newton-Raphson à n dimensions. Typiquement, le problème posé est du type :

$$\begin{cases} f_1(x_1, x_2, ..., x_n) = 0 \\ f_2(x_1, x_2, ..., x_n) = 0 \\ . \\ . \\ . \\ f_n(x_1, x_2, ..., x_n) = 0 \end{cases} \tag{B.2}$$

Au voisinage d'un vecteur $X = (x_1, x_2, ..x_n)$, , chacune des fonctions f_i peut être approximée par son développement de Taylor à un ordre donné. Si l'on considère une approximation à l'ordre 1, nous obtenons :

$$f_i(X + \partial X) = f_i(X) + \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}(X)\delta x_j + O(||\delta X||^2) \tag{B.3}$$

Le principe de la méthode de Newton-Raphson repose alors sur les hypothèses suivantes :
- le vecteur X n'est pas très éloigné de la solution cherchée,
- on cherche alors δX de sorte que $X + \delta X$ se rapproche encore de la solution,
- on néglige tous les termes au-delà du second ordre dans le développement de Taylor ,
- on itère le processus jusqu'à ce que le terme correctif δXsoit assez faible.

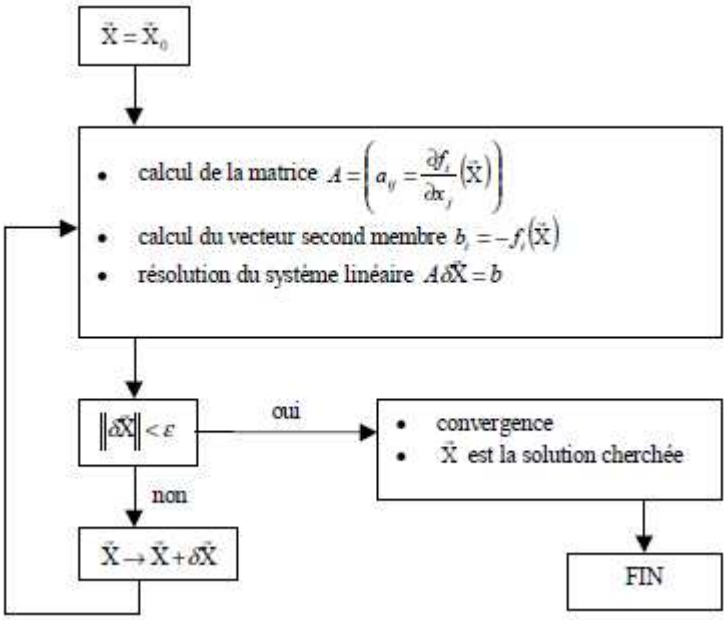

FIGURE B.2 – méthode

Il en résulte alors le système d'équations suivant :

$$f_i(X+\delta X) = f_i(X) + \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}(X)\delta x_j = 0 \qquad donc \qquad \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}(X)\delta x_j = -f_i(X) \tag{B.4}$$

On obtient alors un système linéaire de n équations à n inconnues dont la solution correspond aux composantes du vecteur δX. Le système peut alors être solutionné par des méthodes classiques adaptées à ce genre de problème. L'organigramme suivant détaille les principales phases de la méthode de Newton-Raphson.

Bibliographie

[1] Universal mobile telecommunication system, european telecommunications standards institute. *ETSI UMTS TM*, June 2000.

[2] Satellite digital video broadcasting of second generation (dvb-s2). *ETSI Standard EN302307*, Feb 2005.

[3] Physical layer and management parameters for 10 gb/s operation, type 10gbase-t. *802.3 Working Group*, Sep 2006.

[4] Digital video broadcasting (dvb) ; interaction channel for satellite distribution systems. *European Telecommunications Standards Institute*, July 2007.

[5] Framing structure, channel coding and modulation for satellite services to handheld devices (sh) below 3 ghz,. *Digital Video Broadcasting group*, July 2007.

[6] 3rd generation partnership project - technical specification group radio access network- high speed downlink packet access (hsdpa)-overall description. *3rd Gener. Partnership Project*, Sept 2009.

[7] Evolved universal terrestrial radio access (e-utra). *3rd Gener. Partnership Project 2*, June 2009.

[8] Ieee standard for local and metropolitan area network : Air interface for broadband wireless access systems, ieee computer society. *IEEE Std 802.16TM*, May 2009.

[9] Local and metropolitan area networks specific requirements part 11 : Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 5 : Enhancements for higher throughput. *IEEE 802.11nTM.*, 2009.

[10] R. Rocher A. Chakhari and P. Scalart. Analytical approach to evaluate fixed point accuracy for an iteration of decision operators. In *In Proceedings of the IEEE International Conference on Computer Applications Technology (ICCAT 2013)*, Jan 2013.

[11] W. Luk P. Cheung A. Gaffar, O. Mencer and N. Shirazi. Floating-point bit-width analysis via automatic differentiation. In *International Conference on Field Programmable Logic and Applications, FPL2002,*, pages 158–165, 2002.

[12] L. Gonzalez-Perez A. Morales-Cortes, R. Parra-Michel and T. Cervantes. Finite precision analysis of the 3gpp standard turbo decoder for fixed-point implementation in fpga devices. *in Int. Conf. on Reconfig. Computing FPGAs*, pages 43–48, 2008.

[13] M. Moussa A. Savich and S.Areibi. The impact of arithmetic representation on implementing mlp-bp on fpgas : A study. *IEEE. Transactions on Neural Networks*, pages 240–252, 2007.

[14] T. Adali and S.H. Ardalan. Convergence and error analysis of the Fixed-Point RLS Algorithm with correlated inputs. In *IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP'90)*, pages 1479–1482, 1990.

[15] P. Belanovic and M. Lesser. A library of parameterized floating-point modules and their use. In *International Conference on Field Programmable Logic and Applications, FPL2002*, pages 657–666, 2002.

[16] P. Belanovic and M. Rupp. Fixify : A Toolset for Automated Floating-point to Fixed-point Conversion. In *International Conference on Computing, Communications and Control Technologies (CCCT'04)*, pages 28–32, 2004.

[17] P. Belanovic and M. Rupp. Automated Floating-point to Fixed-point Conversion with the fixify Environment. In *IEEE Rapid System Prototyping (RSP'05)*, pages 172–178, 2005.

[18] F. Berens and N. Naser. Algorithm to system-on-chip design flow that leverages system-studio and systemc. *Synopsys Inc*, 2004.

[19] T. Blankenship and B. Classon. Fixed-point performance of low complexity turbo decoding algorithms. *in IEEE Vehicular Techn. Conf. (VTC)*, 2 :1483–1487, 2001.

[20] T. Bose and M.-Q. Chen. Over flow oscillations in state-space digital filters. *IEEE Transactions on Circuits and Systems*, 38 :807–810, jul 1991.

[21] T. Bose and M.-Q. Chen. Stability of digital filters implemented with two's complement truncation quantization. *IEEE Transactions on Circuits and Systems*, 40 :24–31, jan 1992.

[22] A. Glavieux C. Berrou and P. Thitimajshima. Near shannon limit error correcting coding and decoding : Turbo codes. *in IEEE Intern. Conf. on Commun.*, 2 :1064–1070, May 1993.

[23] C. Douillard C. Berrou, M. Jezequel and S. Keroudan. The advantages of non-binary turbo codes. In *in Inform. Theory Workshop*, pages 61–62, Sep 2001.

[24] P. Cheung C. Ewe and G. A. Constantinides. Error modeling of dual fixed-point arithmetic and its application in field programmable logic. In *International Conference on Field Programmable Logic and Applications, FPL2005*, pages 124–129, 2005.

[25] P. Leong-W. Luk C. H. Ho, C. W. Yu and S. Wilton. Floating-point fpga : Architecture and modeling. *IEEE. Transactions on Very Large Scale Integration Systems*, pages 1709–1718, 2009.

[26] M. Lu C. He, G. Qin and W. Zhao. An efficient implementation of high accuracy finite difference computing engine on fpgas. In *International Conference on Application Specific Systems, Architectures and Processors*, pages 95–98, 2006.

[27] S. Wilton P. Leong C. W. Yu, J. Lamoureux and W. Luk. The coarse-grained/ fine-grained logic interface in fpgas with embedded floating-point arithmetic. *International Journal of Reconfigurable Computing*, pages 1–10, 2008.

[28] J.M. Cheneaux, L.S. Didier, and F. Rico. The fixed cadna library. *Real Number and Computers*, September 2003.

[29] J.M. Cheneaux and J.Vignes. Les fondements de l'arithmétique stochastique. *C.R. Académie des Sciences*, pages 1435–1440, 1992.

[30] Jung-Fu Cheng and T. Ottosson. Linearly approximated log-map algorithms for turbo decoding. *Vehicular Technology Conference Proceedings, 2000. VTC 2000-Spring Tokyo. 2000 IEEE 51st*, 3 :2252 – 2256, May 2000.

[31] G. Constantinides, P. Cheung, and W. Luk. Truncation Noise in Fixed-Point SFGs. *IEEE Electronics Letters*, 35(23) :2012–2014, November 1999.

[32] G. Constantinides, P. Cheung, and W. Luk. Roundoff-noise shaping in filter design. In *IEEE Symposium on Circuits and Systems (ISCAS'00)*, pages IV57–IV60, Geneva, Switzerland, May 28-31 2000. Institute of Electrical and Electronics Engineers.

[33] M. Coors, H. Keding, O. Luthje, and H. Meyr. Integer Code Generation For the TI TMS320C62x. In *International Conference on Acoustics, Speech, and Signal Processing 2001 (ICASSP'01)*, Sate Lake City, US, May 2001.

[34] L. De Coster, M. Ade, R. Lauwereins, and J.A. Peperstraete. Code Generation for Compiled Bit-True Simulation of DSP Applications. In *Proceedings of the 11th International Symposium on System Synthesis (ISSS'98)*, Taiwan, December 1998.

[35] P. Raghavan L. Van der Perre J. Huisken D. Novo, A. Kritikakou and F. Catthoor. Ultra low energy domain specific instruction-set processor for on-line surveillance. *IEEE 8th Symposium on Application Specific Processors (SASP)*, pages 30–35, mar 2010.

[36] L.H. de Figueiredo and J. Stolfi. Affine Arithmetic : Concepts and Applications. *Numerical Algorithms*, pages 1–13, 2003.

[37] C. Douillard E. Boutillon and G. Montorsi. Iterative decoding of concatenated convolutional codes : implementation issues. *Proc. IEEE*, 95 :1201–1227, June 2007.

[38] J. Castura E. Boutillon and F. Kschischang. Decoder-first code design. *in Intern. Symp. on Turbo Codes and Related Topics*, pages 459–462, Sep 2000.

[39] W. J. Gross E. Boutillon and P. G. Gulak. Vlsi architectures for the map algorithm. *IEEE Trans. Commun*, 51 :175–185, 2003.

[40] C. Klein F. de Dinechin and B. Pasca. Generating high-performance custom floating-point pipelines. In *International Conference on Field Programmable Logic and Applications, FPL 2009*, pages 59–64, 2009.

[41] J. Tousch F. Guilloud, E. Boutillon and J.-L. Danger. Generic description and synthesis of ldpc decoders. *IEEE Trans. Commun*, 55 :2084–2091, November 2006.

[42] P. Y. K. Cheung G. A. Constantinides and W. Luk. Wordlength optimization for digital signal processing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32 :1432–1442, 2003.

[43] G. Leyva C. Carreras G Caffarena, J. A. Lopez and O. Nieto Taladriz. Architectural synthesis of fixed-point dsp datapaths using fpgas. *International Journal on Reconfigurable Computing*, pages 1–8, January 2009.

[44] J. A. Lopez G. Caffarena, C. Carreras and A. Fernandez. Sqnr estimation of fixed-point dsp algorithms. *EURASIP Journal on Advanced Signal Processing*, pages 1–21, Feb 2010.

[45] M. Gevers G. Li and Youxian S. Performance analysis of a new structure for digital filter implementation. *IEEE Transactions on Circuits and Systems I : Fundamental Theory and Applications*, 47 :474–482, apr 2000.

[46] R. Gallager. *Low-Density Parity-Check Codes*. PhD thesis, Massachusetts Institutes of Technology, 1960.

[47] D.N. Godard. Self recovering equalization and carrier tracking in two dimensional data communications systems. *IEEE Trans. on Communications*, 28(11) :1867–1875, November 1980.

[48] O. Luthje H. Keding, M. Coors and H. Meyr. Fast bit-true simulation. In *Proceedings of the 38th annual Design Automation Conference*, pages 708–713, 2001.

[49] D. Hocevar. A reduced complexity decoder architecture via layered decoding of ldpc codes. *in IEEE Work. on Signal Proc. Systems, SISP 2004*, pages 107–112, 2004.

[50] http ://reference.wolfram.com/language/.

[51] G. Caffarena J. A. Lopez and C. Carreras. Fast and accurate computation of l2 sensitivity in digital filter realizations. *In Technical Report, University Politechnica de Madrid,*, 2006.

[52] O. Macchi J. Labat and C. Laot. Adaptive decision feedback equalization : can you skip the training period ? *IEEE Trans. on Communication*, pages 921–930, July 1998.

[53] C. Carreras J.A. Lopez, G. Caffarena and O. Nieto-Taladriz. Analysis of limit cycles by means of affne arithmetic computer-aided tests. In *12th European Signal Processing Conference (EUSIPCO 2004)*, pages 991–994, 2004.

[54] C. Carreras J.A. Lopez, G. Caffarena and O. Nieto-Taladriz. Fast and accurate computation of the roundoff noise of linear time-invariant systems. *IET Circuits, Devices Systems*, 2 :393–408, aug 2008.

[55] G. Caffarena J.A. Lopez, C. Carreras and O. Nieto-Taladriz. Fast characterization of the noise bounds derived from coefficient and signal quantization. *In Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, 4 :309–312, may 2008.

[56] F. Cruz-Roldan J.D.O. Campo and M. Utrilla-Manso. Tighter limit cycle bounds for digital filters. *IEEE Signal Processing Letters*, 13 :149–152, mar 2006.

[57] K. Kalliojarvi and J. Astola. Roundoff errors in block-floating-point systems. *IEEE Transactions on Signal Processing*, pages 783–790, apr 1996.

[58] H. Kfir and I. Kanter. Parallel versus sequential updating for belief propagation decoding. *Physica A Statistical Mechanics and its Applications*, 330 :259–270, Dec 2003.

[59] S. Kim, K. Kum, and W. Sung. Fixed-Point Optimization Utility for C and C++ Based Digital Signal Processing Programs. In *Workshop on VLSI and Signal Processing '95*, Osaka, November 1995.

[60] S. Kim and W. Sung. Fixed-Point-Simulation Utility for C and C++ Based Digital Signal Processing Programs. In *Twenty-eighth Annual Asilomar Conference on Signals, Systems, and Computer*, October 1994.

[61] S. Kim and W. Sung. Fixed-Point Error Analysis and Word Length Optimization of 8x8 IDCT Architectures. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(8) :935–940, December 1998.

[62] K. I. Kum and W. Sung. Combined word-length optimization and high-level synthesis of digital signal processing systems. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 20 :921–930, 2001.

[63] F. Jelinek L. Bahl, J. Cocke and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Trans. Inf. Theory*, pages 284–287, Mar 1974.

[64] R. Lauwereins L. De Coster, M. Ade and J. Peperstraete. Code generation for compiled bit-true simulation of dsp applications. In *Proceedings of the 11th International Symposium on System Synthesis 1998*, pages 9–14, dec 1998.

[65] P. Lapsley, J. Bier, A. Shoham, and E. A. Lee. *DSP Processor Fundamentals : Architectures and Features*. Berkeley Design Technology, Inc, Fremont, CA, 1996.

[66] E. E. Lopez A. Bourdoux D. Novo L. Van Der Perre M. Li, B. Bougard and F. Catthoor. Selective spanning with fast enumeration : A near maximum-likelihood mimo detector designed for parallel programmable baseband architectures. In *IEEE International Conference on Communications (ICC) 2008*, May 2008.

[67] M. Mansour and N. Shanbhag. High-throughput ldpc decoders. *IEEE Transactions on VLSI System*, 11 :976–996, Dec 2003.

[68] P. Meignen. Analyse statistique de systémes en virgule fixe. Technical report, ENSSAT, Lannion, Septembre 2005.

[69] D. Menard. *Methodologie de compilation d'algorithmes de traitement du signal pour les processeurs en virgule fixe, sous contrainte de precision*. PhD thesis, Universite de Rennes I, Lannion, Dec 2002.

[70] D. Menard, R. Rocher, P. Scalart, and O. Sentieys. SQNR determination in non-linear and non-recursive fixed-point systems. In *XII European Signal Processing Conference (EUSIPCO'04)*, pages 1349–1352, Vienna, Austria, September 2004.

[71] S.K. Mitra. Digital signal processing laboratory using matlab. *WCB/Mc-GrawHill, University of California, Santa Barbara*, 1999.

[72] G. Montorsi and S. Benedetto. Design of fixed-point iterative decoders for concatenated codes with interleavers. *IEEE J. Sel. Areas Commun*, 19 :871–882, 2001.

[73] A. Oppenheim. Realization of digital filters using block-floating-point arithmetic. *IEEE Transactions on Audio and Electroacoustics*, 18 :130–136, jun 1970.

[74] E. Ozer, A.P. Nisbet, and D. Gregg. Stochastic Bitwidth Approximation Using Extreme Value Theory for Customizable Processors. Technical report, TrinityCollege, Dublin, Ireland, October 2003.

[75] K. Parashar. *System-level Approaches for Fixed-point Refinement of Signal Processing Algorithms*. PhD thesis, Universite de Rennes I, Lannion, Dec 2012.

[76] J. Proakis. *Digital Communications*, volume 1. McGraw-Hill, 4 edition, 2000.

[77] N. Hervé R. Rocher, D. Menard and O. Sentieys. Fixed-point configurable hardware components. *EURASIP Journal on Embedded Systems*, pages 1–13, January 2006.

[78] S. Wilson R. Zarubica, R. Hinton and E. Hall. Efficient quantization schemes for ldpc decoders. *in IEEE Military Commun. Conf. (MILCOM)*, pages 1–5, 2008.

[79] T. J. Richardson and R. L. Urbanke. Efficient encoding of low density parity check codes. *IEEE Trans. Inf. Theory*, 47 :638–656, Feb 2001.

[80] G. Montorsi S. Benedetto, D. Divsalar and F. Pollara. Serial concatenation of interleaved codes : performance analysis, design and iterative decoding. *IEEE Trans. Inf. Theory.*, 44 :909–926, May 1998.

[81] G. Montorsi S. Benedetto, D. Divsalar and F. Pollara. Soft-input soft-output modules for the construction and distributed iterative decoding of code networks. *Europ. Trans. on Telecomm.*, 9, Apr 1998.

[82] R. Serizel. Implantation en virgule fixe d'un codeur audio. Master's thesis, ENSSAT, Lannion, Septembre 2006.

[83] C. Shi. *Floating-point to Fixed-point Conversion*. PhD thesis, University of California, Berkeley, Lannion, Apr 2004.

[84] C. Shi and R. W. Brodersen. Floating-point to fixed-point conversion. *IEEE Trans. Signal Processing*, 2004.

[85] C. Shi and R.W. Brodersen. A perturbation theory on statistical quantization effects in fixed-point dsp with non-stationary inputs. In *Proceedings of the 2004 International Symposium on Circuits and Systems, ISCAS 2004*, volume 3, may 2004.

[86] W. Mecklenbrauker T. Claasen and J. Peek. Effects of quantization and over ow in recursive digital filters. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 24 :517–529, dec 1976.

[87] T. Inoue W. Zeng T. Hinamoto, S. Yokoyama and Wu-Sheng Lu. Analysis and minimization of l2-sensitivity for linear systems and two-dimensional state-space filters using general controllability and observability gramians. *IEEE Transactions on Circuits and Systems I : Fundamental Theory and Applicationsn*, 49 :1279–1289, sep 2002.

[88] Z. Wang T. Zhang and K. Parhi. On finite precision implementation of low density parity check codes decoder. *in Proc. IEEE ISCAS*, 4 :202–205, May 2001.

[89] R. Tanner. A recursive approach to low complexity codes. *IEEE Trans. Inf. Theory*, 27 :533–547, Sep 1981.

[90] C. Mullis W. Mills and R. Roberts. Digital filter realizations without overflow oscillations. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26 :334–338, aug 1978.

[91] S. Wadekar and A. Parker. Accuracy Sensitive Word-Length Selection for Algorithm Optimization. *IEEE/ACM International Conference on Computer Design (ICCAD'98)*, pages 54–61, 1998.

[92] B. Widrow and I. Kollar. Quantization noise : Roundoff error in digital computation, signal processing, control, and communications. *Cambridge University Press, Cambridge, UK*, 2008.

[93] B. Widrow, I. Kollár, and M.-C. Liu. Statistical Theory of Quantization. *IEEE Transactions on Instrumentation and Measurement*, 45(2) :353–361, Apr. 1996.

[94] Xilinx. System generator for dsp. In *http ://www.xilinx.com*.

[95] H. Zhong and T. Zhang. Block-ldpc : A practical ldpc coding system design approach. *IEEE Trans. Circuits Syst*, 52 :766–775, Apr 2005.

[96] U. Zölzer. *Digital Audio Signal Processing*. John Wiley and Sons, August 1997.